Australia A $4.75 | New Zealand NZ $ 6.50
Singapore S $9.45 | Hong Kong H $23.50
Malaysia M $9.45 | Sweden 30 SEK

$2.95 USA

68' MICRO JOURNAL

* Motorola 68020 *
6809-68008-68000-68010

VOLUME VIII ISSUE VIII • Devoted to the 68XX User • August 1986

"Small Computers Doing Big Things"

0 74470 12810 08

WE HAVE MOTOROLA COVERED

GESSBS-4 $316*

1 MHz 6809 CPU
Sockets for up to 32 Kbytes EPROM
Sockets for up to 16 Kbytes CMOS RAM
One RS 232 serial port
40 TTL Bidirectional I/O lines
4 x 16-bit timers

GESMPU-14 $636*

8 MHz 68010 CPU
Optional 32081 arithmetic unit
Sockets for up to 128 Kbytea EPROM
One RS 232 aerial port
4 x 8-bit timers
Real-time clock/calendar and battery

GESMPU-4A $316*

8 MHz 68000 CPU
Sockets for up to 128 Kbytes EPROM
Sockets for up to 64 Kbytes CMOS RAM
One RS 232 serial port
Three 18-bit timers

SOFTWARE

OS-9®, PDOS®, CP/M 68K®, Editor-Assembler, Basic-Pascal-C compilers, FORTH.

Gespac

USA - CANADA
100 West Hoover Ave.
Mesa, AZ 85202
Tel. (602) 962-5559
Telex 386575

INTERNATIONAL
3,chemin des Aulx
CH-1228 Geneva
Tel. (022) 713400
Telex 429989

★ 100 piece quantities

Ⓜ is a registered trademark of Motorola Inc.

GMX Micro-20 prices

MICRO 20 (12.5 MHz).....$2565.00
MICRO 20 (16.67 MHz).....$2895.00
8 PORT RS232 BOARD SET (SBC-8S).....$ 498.00
PROTOTYPING BOARD (SBC-WW).....$ 75.00
BACK PANEL PLATE (BPP-PC).....$ 44.00
I/O BUS ADAPTER (SBC-BA).....$ 195.00

QUANTITY DISCOUNTS ARE AVAILABLE ON THE
ABOVE ITEMS AS FOLLOWS:4-9,LESS 5%;
10-24,LESS 10%; 25-99,LESS 20%; 100 UP,LESS 30%.

MC6888IRC12.....$ 295.00
MC6888IRC16.....$ 395.00
SBC ACCESSORY PACKAGE (M20-AP).....$1690.00
for other configurations and options, contact GMX.
MOTOROLA 68020 USERS MANUAL.....$ 18.00
MOTOROLA 68881 USERS MANUAL.....$ 18.00

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 6 weeks for collection so we would advise wiring money, or checks drawn on a Bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

GMX
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

GMX S-50 BUS prices

68020 SYSTEM

For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer, DMA transfers, high speed MMU, we have the UNIFLEX-VM 68020 development system.
The system CPU provides protection to the system and other users from crashes caused by defective user programs.
The system's intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions.
The UniFLEX VM Operating System is a demand-paged, virtual memory operating system written in 68020 Assembler code for compactness and efficiency. It allows up to 4 Megabytes of Virtual Memory per user. All systems include 1MB of static RAM, one 3-port Intelligent Serial I/O board, DMA Controllers, & 5" 8M track floppy drive.

PRICES

#020 UniFLEX VM with 25MB HD.....$10,988.20
#020 UniFLEX VM with 85MB HD.....$12,488.20

YOU CAN EXPAND THESE 020 SYSTEMS WITH:
60MB STREAMER.....$ 2,400.00
REMOVABLE PACK DRIVE.....$ 1,200.00

INTELLIGENT I/OS
#14 3 Port Serial-30 Pin.....$ 498.14
#13 4 Port Serial-50 Pin.....$ 618.13
#12 Parallel-50 Pin.....$ 538.12

CABLE SETS FOR I/OS
95 Cable Sets Specify Card.....$ 24.95
51 Cent. D.P. Cable for #12 & #44....$ 34.51
53 Cent. Cable Set.....$ 36.53

6809 SYSTEM

The number 39 systems include:#05 CPUwDAT; #19 Classy Chassis; 256K Static RAM; a # 43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates;

System # 39 OS-9 GMX II Dual 8M 8SDD...$ 2,998.39
" w19MB.....$ 4,698.39
" w72MB.....$ 6,298.39

The Software included in this System:
GMXBUG monitor; FLEX; and OS-9 GMXII. You can software select either FLEX or OS-9. Also includes OS-9 Editor, Assembler, Debugger, BASIC-09, RUNB, RMS, DD, and GMX-VU1SK for FLEX.

System # 39 UniFLEX w25MB.....$ 4,698.39
" w85MB.....$ 6,298.39

The UniFLEX Operating System is included.

6809 SYSTEMS USING THE GIMIX III CPU & INTELLIGENT I/O PROCESSOR BOARDS

These System include: GMX6809 CPU III; one #11 3 port intelligent serial I/O & Cables; #19 Classy Chassis; 256K Static RAM; #68 DMA controller; all necessary cables, power regulators, and filler plates.

System # 79 OS9 GMX III Dual 8M OSDD...$ 4,498.79
" w25MB.....$ 6,498.79
" w85MB.....$ 7,998.79

The # 79 System Software includes: OS9 GMXIII; OS9 Editor, Assembler, Debugger, BASIC 09, RUNB, RMS, DD, RAMdisk, O-FLEX; GMXBUG; FLEX. The GMX Support ROM and the hardware CRC board are exclusive features included in this system.

System # 89 UniFLEX III w25MB.....$ 6,798.39
w85MB.....$ 8,298.39

The UniFLEX GMX III Operating System is included.

68' MICRO JOURNAL

Portions of the text for '68' Micro Journal were prepared using the following furnished Hard/Software:

COMPUTERS – HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, TX 78216
S.9 - 5:8 DMF Disk - CDS1 - 8212W - Sprint 3 Printer

GMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe - OS9 - FLEX - Assorted Hardware

EDITORS – WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX - Editor - Text Processor

Stylo Software Inc.
PO Box 916
Idaho Falls, ID 83402
Stylograph - Mail Merge - Spell

Contents

Send All Correspondence To:

Computer Publishing Center
68' Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Phone (615) 842-4600 or Telex 5106006630

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, TN and additional entries. Postmaster: send form 3597 to 68' Micro Journal, POB 849 Hixson, TN 37343.

Subscription Rates

1 Year $24.50 U.S.A., Canada & Mexico Add **$9.50** a Year. Other **Foreign Add $12 a Year** for **Surface, Airmail Add $48 a Year.** Must be in **U.S. currency!!**

Items or Articles For Publication

Articles submitted for publication should include authors name, address, telephone number and date. Articles should be **on either 5 or 8 inch disk** in STYLOGRAPH or TSC Editor format with **3.5 inch column width.** All disks will be returned. Articles submitted **on paper** should be **4.5 inches in width** (including Source Listings) for proper reductions. **Pleaaa Use A Dark Ribbon!!** No Blue Ink!!! Single space on 8x11 bond or better grade paper. No hand written articles accepted. Disks should be in FLEX2 6800 or FLEX9 6809 any version or OS-9 any version.

The following TSC Text Processor commands **ONLY** should be used: **.sp** space, **.pp** paragraph, **.fl** fill and **.nf** no fill. Also please **do not format** within the text with **multiple spaces.** We will enter the rest at time of editing.

All STYLOGRAPH commands are acceptable except **.pg** page command. We print edited text files in continous text form.

Letters To The Editor

All letters to the editor should comply with the above requirements and **must be signed.** Letters of "gripes" as well as "praise" are solicited. We reserve the right to reject any submission for lack of "good taste" and we reserve the right to define "good taste".

Advertising Rates

Commercial advertisers please contact 68' Micro Journal advertising department for current rate sheet and requirements.

Classified Advertising

All classified ads must be non-commercial. Minimum of **$9.50 for first 20 words** and .45 per word after 20. All classifieds must be **paid in advance.** No classified ads accepted over the phone.

The VME BUS and OS-9:

Ultimate Software for the Ultimate Bus.

Modularity. Flexibility. High Performance. Future growth. These are probably the prime reasons you chose the VME bus. Why not use the same criteria when selecting your system software? That's why you should take a look at Microware's OS-9/68000 Operating System—it's the perfect match for the VME bus.

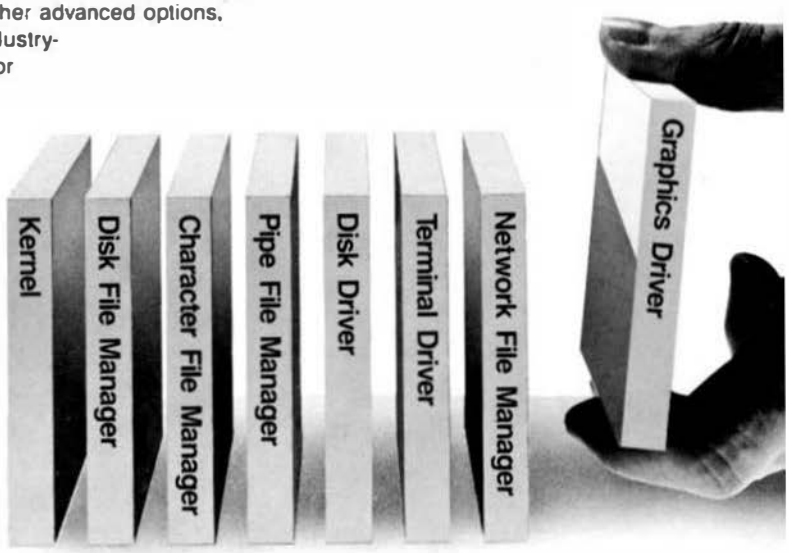
When you're working with VME you must have access to every part of the system. Unlike other operating systems that literally scream KEEP OUT!, OS-9's open architecture invites you to create, adapt, customize and expand. Thanks to its unique modular design, OS-9 naturally fits virtually any system, from simple ROM-based controllers up to large multiuser systems.

And that's just the beginning of the story. OS-9 gives you a complete UNIX-application compatible environment. It is multitasking, real time, and extremely fast. And if you're still not impressed, consider that a complete OS-9 executive and I/O driver package typically fits in less than 24K of RAM or ROM.

Software tools abound for OS-9, including outstanding Microware C, Basic, Fortran, and Pascal compilers. In addition, cross C compilers and cross assemblers are available for VAX systems under Unix or VMS. You can also plug in other advanced options, such as the GSS-DRIVERS™ Virtual Device Interface for industry-standard graphics support, or the OS-9 Network File Manager for high level, hardware-independent networking.

Designed for the most demanding OEM requirements, OS-9's performance and reliability has been proven in an incredible variety of applications. There's nothing like a track record as proof: to date, over 200 OEMs have shipped more than 100,000 OS-9-based systems.

Ask your VME system supplier about OS-9. Or you can install and evaluate OS-9 on your own custom system with a reasonably priced Microware PortPak™. Contact Microware today. We'll send you complete information about OS-9 and a list of quality manufacturers who offer off-the-shelf VME/OS-9 packages.



Modular Hardware Deserves Modular Software

MICROWARE.

Microware Systems Corporation
1866 N.W. 114th Street • Des Moines, Iowa 50322
Phone 515-224-1929 • Telex 910-520-2535

Microware Japan, Ltd.
41-19 Honcho 4-Chome, Funabashi City • Chiba 273, Japan • Phone 0473 (28) 4493 • Telex 781-299-3122

Micromaster Scandinavian AB
S:t Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
Phone: 018-138595
Telex: 76129

Dr. Rudolf Keil, GmbH
Porphystrasse 15
D-6905 Schriesheim
West Germany
Phone: (0 62 03) 67 41
Telex: 465025

Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83-3377
Telex: 828275

Vivaway Ltd.
36-38 John Street
Luton, Bedfordshire, LU1 2JE
United Kingdom
Phone: (0582) 423425
Telex: 825115

Microprocessor Consultants
92 Bynya Road
Palm Beach 2108
NSW Australia
Phone: 02-919-4917

Microdata Soft
97 bis, rue de Colombes
92400 Courbevoie
France
Phone: 1-768-80-80
Telex: 615405

MUSTANG -020 Super SBC™

DATA-COMP proudly presents the first Under $5000 "SUPER MICRO".

The MUSTANG-020™

MUSTANG-020™

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $5000, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 20 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The system SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $3000. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based bebugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

DATA-COMP Installed Systems World-Wide — OVER 10 YEARS OF DEDICATED QUALITY!

CPI A Division of Computer Publishing, Inc.

5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

MUSTANG-020 Benchmarks **
Time Seconds

Type System	32 bit Int. Loop	Register Long Loop
IBM AT 7300 Xenix Sys 3	9.7	No Registers
AT&T 7300 UNIX PC 68010	7.2	4.3
DEC VAX 11/780 UNIX Berkley 4.2	3.6	3.2
DEC VAX 11/750 " " "	5.1	3.2
68000 OS9 68K 8 Mhz	18.0	9.0
68000 " " 10 Mhz	6.5	4.0
MUSTANG-020 68020 MC68881 OS9 16 Mhz	2.2	0.88
MUSTANG-020 68020 MC68881 UNIFLEX "	1.8	1.22

** Loop:
```
Main()
{
  register long i;
  for (i=0; i < 999999; ++i);
}
```

Estimated MIPS - MUSTANG-020 - 2.5 MIPS
Motorola Specs: Burst up to 7 - 8 MIPS - 16 Mhz

MUSTANG-020-

FEATURES

- 12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path processor
- 32-bit wide data and address buses, non-multiplexed
- on chip instruction cache
- object code compatible with all 68XXX family processors
- enhanced instruction set - math co-processor interface
- 68881 math hi-speed floating point co-processor (optional)
  direct extension of full 68020 instruction set
  full support IEEE P754, draft 10.0
  transcendental and other scientific math functions
- 2 Megabyte of SIP RAM (512 x 32 bit organization)
- up to 256K bytes of EPROM (64 x 32 bits)
- 4 Asynchronous serial I/O ports standard
  optional to 20 serial ports
  standard RS-232 interface
  optional network interface
- buffered 8 bit parallel port (1/2 MC68230)
  Centronics type pinout
- expansion connector for additional I/O devices
  16 bit data path
  256 byte address space
  2 interrupt inputs
  clock and control signals
  Motorola I/O Channel Modules
- time of day clock/calendar w/battery backup
- controller for 2, 5 1/4" floppy disk drives
  single or double side, single or double density
  35 to 80 track selectable (48-96 TPI)
- SASI interface
- programmable periodic interrupt generator
  interrupt rate from micro-seconds to seconds
  highly accurate time base (5 PPM)
- 5 bit sense switch, readable by the CPU
- hardware single-step capability
- mounts directly to a standard 5 1/4" disk drive

Size 8 15/16 x 5 7/8

These *hi-speed* 68020 systems are presently working at NASA, Atomic Energy Commission, other Government Agencies as well as Universities, Business, Labs, and critical applications centers, Worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a *must*!

MUSTANG-020™ Software

OS-9

OS-9	$350.00
Basic09	300.00
C Compiler	400.00
Fortran 77	400.00
Microware Pascal	400.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX	$450.00
Screen Editor	150.00
Sort-Merge	300.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

8 Port expansion RS-232 498.00
(total of 20 serial ports supported)

Expansion for Motorola I/O Channel Modules $195.00

Sculptor+: We are USA distributors for Sculptor+. Call or write for site or multiple system license & discounts, OEM/Dealer.

Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. Save $300.00!

Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information.

MUSTANG-020 System component prices - Effective July 1, 1986
Prices subject to change - call for latest quotes.

MUSTANG-020 (12.50 Mhz)		$2750.00
Cabinet		$299.95
5"-80 track floppy	DS/DD	$269.95
Floppy cable		$39.95
OS-9 68K		$350.00
Winchester cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Xebec H/D controller		$395.00
Shipping USA UPS		$20.00
Total:		$5059.80

DISCOUNT LIMITED TIME: with Complete System $1061.00

Complete System $3998.80

OPTIONS ADD:

UniFLEX	$90.00
MC68881 f/p math processor	$275.00
16.67 Mhz MC68020	$375.00
16.67 Mhz MC68881	$375.00

WE WILL NOT BE UNDERSOLD!

This price subject to increase
Additional MUSTANG systems soon

Note: Current OS-9 (Ver. 1.2) does not address the MC68881 - Future revisions will. If the 68881 is anticipated in the future, it must be ordered with the system, when originally ordered. UniFLEX does support both the enhanced code of the 68020 and 68881 now.

DATA-COMP
10 YEARS OF DEDICATED QUALITY!

PAT - JUST

PAT

With 'C' Source

$229.00

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

SOFTWARE USERS NOTES

BY: Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

FOR THOSE WHO
68 MICRO JOURNAL™
NEED TO KNOW!

I am going to take some space this month to comment on the FFT programs in the April issue of '68'. First of all, let me say that I was very happy to see the programs and the good discussion of them. I noted the time for the Uhrich algorithm, which was immediately interesting because we had implemented the Cooley Tukey algorithm, and it would appear that the Uhrich will run three times as fast! I entered the program in Pascal on the MUSTANG-020™ system and compiled it with OmegaSoft Pascal. I found the execution time to be about 4.5 seconds for a 1024 point FFT. The article by James Gross reports that it ran 132 seconds on a 2 MHz 6809 system in Lucidata Pascal. I decided to see what I could do to speed up the program some.

The first try was to generate a table of sines for the first quadrant (256 values for 0 to 90 degrees). I was lazy so I wrote a quick BASIC program to generate the table to a file, and then edited it into the program file with PAT. Rather than generate a whole half cycle table for Sine and another for Cosine, I wrote a "fold" procedure for sine and cosine to allow me to use a quarter cycle table. I recompiled and found that the program now executed in just about 2.5 seconds. Looking a value up in a table is always faster than a calculation involving floating point numbers, even more so one that involves a scientific function.

Now I decided to see what else could be done. The original program is nicely illustrative of the algorithm, and my comments here are NOT to be taken as being critical of the article, the purpose of which was to show the algorithms clearly and not to optimize the program for speed. First of all, to speed up a program it is necessary to remove all calculations from within a loop that can be done outside of the loop. For example the W := 2*pi/N; calculation contains no variables that change with any loop count, so that calculation can be done just once for a run of the program. Of course with the table, that calculation is not required. Several other values were calculated repeatedly inside of the I loop, and needed to be calculated only once for each time J was incremented, so these were pulled out of the loop. Several places in the I loop, calculations were done involving a divide by N. At least in the 6809 software, a divide costs a lot more time than a multiply. This is primarily so because the 6809 has the MUL instruction (an 8 bit by 8 bit multiply with a 16 bit result, in 12 clock cycles). It has no equivalent divide instruction. The 8 by 8 multiply doesn't sound very useful, but it can be used as a building block to make much larger multiple precision multiply routines. Rather than divide by N it is much faster to calculate 1/N once somewhere in the program and then to multiply by that value. Several of the calculations contained expressions involving the same variables, and these expressions could be evaluated once and assigned to another variable to be used within the loop.

At about that time I decided to code the algorithm in PL/9 and run it on the 2MHz 6809 system. It ran about 20 seconds. I did some further work with the 6809 version in PL/9, much of which was transferrable back to the Pascal version.

One place that uses up a lot of time is the testing of the loop condition. FOR I := 0 TO N/4, for example causes the value N/4 to be calculated each time through the loop. Since for a given program N is constant, the value N4 = N/4 was made a constant, and the substitution made. Similarly, most of the DIV operations and the REAL divides (/) were eliminated.

I noted that each time through the J loop, the program operates on the values in the X[1,N] arrays and puts the results in the X[2,N] arrays. Later the contents of the 2 array is put back into the 1 array for the next time through the J loop. I decided that rather than move the result back to the

first array after each pass, I would simply decide whether the pass (or the count in J) were odd or even, and "shuttle" the values back and forth between the two arrays. For an even value for NSTAGE, this does not complicate the output procedure. For an odd NSTAGE, it would leave the result in the 2 array, requiring some further logic to unscramble it. Since I was optimizing for N=10, I left things as they were. Next I saw in the SIN_FOLD and COS_FOLD procedures in the logic that determines whether the angle is in the first or second quadrant, that I had divided INDEX by N4, a better operation than my first attempt which was 4*INDEX DIV N;. I decided that the PL/9 SHIFT operation could be used to advantage. SHIFT(INDEX -8) would do nicely for the 1024 point FFT, and a generalization would be SHIFT (INDEX,-(NSTAGE-2)). I defined another constant NSM2 with that value. Later I decided that INDEX AND 256 would work as well and be considerably faster. The final results are the SIN_FOLD and COS_FOLD procedures as they appear in the listing.

At this point, the result ran on the 6809 system in just about 14.4 seconds, and on the MUSTANG-020™ in Pascal in 1.8 seconds. Note that our MUSTANG-020™ does NOT have the arithmetic Co-processor. About the only improvement that I couldn't transfer back to Pascal was the SHIFT operation. C has this operation but Pascal does not. The listings for both the Pascal and the PL/9 versions are included here.

A few rules should be evident. **1.** Don't calculate anything more than once for each time the result of the calculation will change (an index is incremented etc.). **2.** If a value doesn't change for the whole operation of the program, define it as a constant. **3.** Use tables for values rather than calculating them. It is always faster to calculate an array index than to evaluate a REAL expression. **4.** A program optimized for execution speed is likely to be more obscure than one optimized for clarity and flexibility.

A further note is that I found a bug in the Uhrich algorithm as presented in April '68'. I was able to find and fix it, but it added about 40% to the calculations performed. Final result was 2.6 seconds for the 1024 point FFT. At that point, the 6809 version stretched its execution time back out to 23 seconds. I decided to try a 16 bit integer version with an integer sine table and a special multiply routine that would treat the sine value between 0 and 32767 as a fraction between 0 and 1.0. After several tries, the final result was a program that does the 1024 point FFT in 3.7 seconds on a 2 MHz 6809 system. The listing of this program is included here.

Woes of a Multilingual Programmer

Since the arrival of the MUSTANG-020™, I've spent a lot of time writing C programs and some considerable time again in Pascal, which I have not used for some time. Now I am all confused. I write Pascal If-Then statements without the THEN. I include the THEN in a C program. I use := in a PL/9 program and = in a Pascal program. I forget the == in a C program test if(a==b). Tonight I tried to use the C shift (>>) function in a Pascal program. Maybe someday I will reach the point of being able to switch comfortably, but I certainly am not there yet. Oh, yes, one more irritation, when I write a BASIC program now, I try to put a semicolon at the end of every line! Of course I like them all. If I didn't I wouldn't have these troubles. Part of the fun of computing is having something to complain about! How about a few examples?

```
if (a==b) statement;
  else differentstatement;

if a=b then statement
   else differentstatement;
```

The first example is the C version of an IF THEN ELSE. Note that the word THEN is not used. The semicolon after the IF clause is REQUIRED. The second example is the Pascal or PL/9 version. In Pascal, the THEN clause must NOT have a semicolon after it. PL/9 doesn't seem to care if it is there or not!

```
while (a<b)
{ statement;
  statement;
}

While a < b do
begin
  statement;
  statement
end;
```

Again the first example is the C version. The condition goes inside parentheses and the word DO is omitted. Both statements must have a semicolon. The second example is the Pascal version. Parentheses are not used around the condition. DO is used. The last statement before the END does not need a semicolon, and in fact the Pascal standard indicates that one should not be used there, but most Pascal compilers don't care. I find that if I leave it out, I later add a statement after the

one without the semicolon and forget to add the semicolon, so I get a compile error. PL/9 uses the same format as Pascal but without the DO.

```
do
{ statement;
  statement;
} while (a<b);
```

```
repeat
  statement;
  statement
until a >=b;
```

Again the C version is first. Note that the test at the end of the loop is a WHILE. That is the loop repeats until the condition becomes FALSE. In the Pascal version the condition is an UNTIL. That is, the loop repeats until the condition becomes TRUE. In translating a program from one to the other, you must be sure to "reverse" the condition. Less Than becomes not(Less Than) which is the same thing as Greater Than or Equal. The PL/9 version of this loop is the same as the Pascal version.

Case statements are different in the three languages too. The differences are small, but just enough to be confusing and to send me to the manual every time I switch. My problem is not that the constructs of these languages are different, but that they are too similar. The differences are so small that it is easy to forget them. The Pascal standard is for the program to be written in upper case only. Some Pascal compilers allow lower case for user defined variables, but insist on upper case for keywords. Some now allow all lower case as a user choice. C requires lower case for keywords, and the C "convention" is to use lower case for user variables and upper case for CONSTANTS. PL/9 started out as an upper case only language, but now allows upper and lower, and does not distinguish between them.

Reader Feedback

Some indirect feedback arrived via the letter in April '68' from Calvin Dodge. I won't speak for the other items that he mentioned, but I will speak for the parts of my column on which his letter contained comments. First of all, Calvin, you are absolutely correct with your first comment about my simplest and fastest CLEAR memory loop in the March column. The code was:

```
  clra
  clrb
loop std ,--x
  bne loop
```


You pointed out correctly that the loop will terminate immediately because the data in ACCD when the STD instruction is executed, is what sets the zero flag, not the post decrement of X. All I can say is that I vow never again to publish ANY assembler code no matter how small or trivial, without testing the idea first! Many programmers, myself included, tend to see what they want the code to do rather than what it actually does. I plead guilty! I think (breaking my vow of the last paragraph) the following would work correctly, though not quite as efficiently:

```
  clra
  clrb
loop std 0,x
  leax -2,x
  bne loop
```

the leax instruction does set the zero flag when the value in the X register reaches zero.

A little later in your letter you refer to my bit on code efficiency in PL/9. I'm afraid you missed the point. PL/9 has a logical .AND and a bitwise AND. The difference is that the logical .AND has the period in front of it. My point was that the seemingly less straightforward coding:

```
IF A=B THEN IF C=D...
```

produced considerably less compiler output (object code) than the more straightforward (IF A=B .AND C=D) logical .AND version did. The reason for that must lie in how the code generation is carried out, and it is surely a peculiarity of PL/9.

Your suggestion on the ADTEST code is certainly a valid improvement, and its use extends to other values for multiplication by a constant just as well. We find at work that passing a program back and forth between a couple of programmers results in major improvements. Each sees a way to improve on the code of the other. A substantial part of the improvement in the FFT program execution time discussed above, is the result of two of us passing the ideas back and forth several times. When I thought I had gone about as far as I could go, another programmer, Doug Case made a few

more suggestions, and by the time we were both through making further changes, we had taken another 10% of the execution time out of the algorithm. Since the program spends just about half of its time multiplying two numbers, the point at which removal of execution cycles will make the most improvement is in the multiply routine. For an hour or so, each suggestion by one of us brought a further suggestion from the other. Eventually we reached a point where further changes would not significantly improve the timing. Anyway, Calvin, thanks for keeping me honest!

By the way, by now you all noticed that the June issue was missing my column. A number of difficulties had come up that prevented me from finishing a column by the deadline. That was my first miss in very nearly six years of writing for this Journal. I am in the process now, of trying to get ahead a little bit, so that if I again get too involved in other things, it won't make me miss another deadline.

PC Boards

For the last couple of days at work, I have been working on a PC board layout using our new Tandy 3000 and the Wintek SmArtwork package. We traded one of our Tandy 1200-HD computers in on the newer 3000 with the 80286 processor. SmArtwork is much more pleasurable to use on the 3000, which seems to run about 5 times faster, though after a couple of days, one tends to see red, green, and yellow spots in his dreams at night. At my present rate, it will take about three more days to finish the card and it will cost us about one third as much as previous such cards that were done outside by a PC layout company. Actually, the savings in cost on this card and a similar one, will be enough to pay for the new computer.

One of the nicest things about doing a PC board on the computer, is the ease with which changes may be made as the layout progresses. If you use a little care in the placement of the I.C. packages, the design progresses rapidly, and if you box yourself in and can't get to a connection, it is easy to delete wires and make changes to get out of a corner. We have been using the Wintek software for over a year now, and they have continued to upgrade it and add new features. The latest version has the possibility of four different line widths, filling in solid areas, and four different kinds of pads, two round and two square. Several of the early inconveniences in editing a layout have been removed.

Though I realize that this discussion is out of the realm of this column's title, I highly recommend this software to anyone who wants to design printed circuit boards. It is not the ultimate in automatic PC board layout, but the user can insert or move a portion of the layout (a BLOCK). He can let wires autoroute, though the feature tends to spread conductors out too much, since the router can't know how many more conductors must fit a particular space. We tend to "guide" the placement of conductors by incrementally routing them more or less where we want them. At the end when we are faced with a few impossible connections, we turn the auto router loose and it can frequently find a route, upon which we can improve by means of feed-throughs and running the conductor on both sides of the board.

+++

```
PROGRAM FAST(INPUT, OUTPUT);

CONST
   N=1024;
   ONE_N = 1.0/N;          { 1/N  }
   NM1 = N-1;              { N-1  }
   N2 = N DIV 2;           { N/2  }
   N4 = N DIV 4;           { N/4  }
   TWO_N = 2.0/(N-1);      { 2/(N-1)  }
   NSTAGE = 10;            { POWER OF 2 OF N }
   NSM2 = -8;              { -(NSTAGE-2)  }
   PI = 3.14159265;

TYPE
   COMPLEX = RECORD
      RE : REAL;
      IM : REAL
   END;

VAR
   TABLE : ARRAY[0..N DIV 4] OF REAL;
   P2 : ARRAY[0..NSTAGE+1] OF INTEGER;
   X : ARRAY [1..2.0..NM1] OF COMPLEX;


{ THE FOLLOWING SETS UP A TABLE OF SINES.  A COMING VERSION OF
  ORS PASCAL WILL ALLOW INITIALIZATION OF A TABLE OF CONSTANTS
  IN A PROGRAM SO THIS WILL NOT BE NECESSARY  }

PROCEDURE INITARRAY;
   VAR I,J:INTEGER;
BEGIN
   FOR I:= 0 TO N4 DO TABLE[I] := SIN(2*PI*I/N);
   J:=1;
   FOR I:=0 TO NSTAGE+1 DO
   BEGIN
     P2[I]:=J;
     J:= J*2;
   END;
END;


FUNCTION SIN_FOLD(INDEX:INTEGER):REAL; { QUADRANT SORTING LOGIC }
BEGIN
   IF INDEX AND 256 <>0
```

```
      THEN SIN_FOLD := TABLE[N2-INDEX]
      ELSE SIN_FOLD := TABLE[INDEX];
END;


FUNCTION COS_FOLD(INDEX:(INTEGER):REAL; { QUADRANT SORTING LOGIC }
BEGIN
   IF INDEX AND 256 <>0
      THEN COS_FOLD := -TABLE [INDEX -N4]
      ELSE COS_FOLD := TABLE [N4-INDEX];
END;

PROCEDURE INPUT_FUNCTION;  {SAWTOOTH 1.0 PEAK AMPLITUDE }
VAR I: INTEGER;


BEGIN
  FOR I := 0 TO MAX DO
  BEGIN
     X[I,I].RE := -1.0 + 1.0/NO_N;
     X[I,I].IM := 0.0;
  END;
END;

PROCEDURE FASTFOURIER;

VAR A,B,C,D,E,F,J: INTEGER;
    I,A,BN2J,NOP2,P2D2I INTEGER;
    W: COMPLEX;
    SINW, COSW : REAL;
    JODD : BOOLEAN;

BEGIN
   FOR J := 1 TO NSTAGE DO
   BEGIN
      IF ODD(J) THEN JODD := TRUE ELSE JODD := FALSE;
      NOP2 := N DIV P2[J];
      P2D2 := P2[J] DIV 2;

      FOR I := 0 TO P2D2-1 DO
      BEGIN
         IN2J := I*NOP2;
         COSW := COS_FOLD[IN2J];
         SINW := SIN_FOLD[IN2J];
         B := IN2J;
         F := 0*2;
         FOR A := 0 TO NOP2-1 DO
         BEGIN
            A := A+B;
            B := A+F;
            C := B+NOP2;
            E := A+N2;
            IF JODD THEN
            BEGIN
               W.RE := X[1,C].RE*COSW + X[1,C].IM*SINW;
               W.IM := X[1,C].IM*COSW -X[1,C].RE*SINW;
               X[2,A].RE := X[1,B].RE + W.RE; { **--  }
               X[2,A].IM := X[1,B].IM + W.IM;
               X[2,E].RE := X[1,B].RE - W.RE;
               X[2,E].IM := X[1,B].IM - W.IM;
            END
            ELSE
            BEGIN
               W.RE := X[2,C].RE*COSW + X[2,C].IM*SINW;
               W.IM := X[2,C].IM*COSW - X[2,C].RE*SINW;
               X[1,A].RE := X[2,B].RE + W.RE;
               X[1,A].IM := X[2,B].IM + W.IM;
               X[1,E].RE := X[2,B].RE - W.RE;
               X[1,E].IM := X[2,B].IM - W.IM;
            END;
         END;
      END;
   END;
END;

FUNCTION MAG(X,Y:REAL):REAL;
BEGIN
   MAG := SQRT(SQR(X) + SQR(Y));
END;
```

FOR THOSE WHO 68 MICRO JOURNAL™ NEED TO KNOW!

```
PROCEDURE OUTPUT;
VAR
   I,J: INTEGER;
   VALUE, VALUE1: REAL;
BEGIN
   VALUE1 := MAG(X[1,1].RE,X[1,1].IM);
   FOR I := 0 TO (N DIV 2) -1 DO
   BEGIN
      VALUE := MAG(X[1,I].RE,X[1,I].IM);
      WRITELN(I:3,VALUE/VALUE1:12:6,
        X[1,I].RE:12:6,X[1,I].IM:12:6);
   END;
  WRITELN;
END;



BEGIN
   INITARRAY;
   INPUT_FUNCTION;
   WRITELN;
   WRITE ('START PROGRAM'); { ADDED FOR TIMING ACTUAL FFT }
   WRITELN;
   FASTFOURIER;
   OUTPUT;
END.

{* FFT USING UNROLLED ALGORITHM *}

ORIGIN = 0;
STACK = $1800;

CONSTANT
   N = 1024,
   N2 =512,
   N4 =256,
   NSTAGE = 10,
   NSM1 = -8;

A: $1800: INTEGER 1IRE:(1024),
                  1IIM:(1024),
                  L2RE:(1024),
                  L2IM:(1024);


   INTEGER TABLE
     0,  201,  402,  603,  804, 1005,
    1206, 1406, 1607, 1808, 2009,
    2209, 2410, 2610, 2811, 3011,
    3211, 3411, 3611, 3811, 4011,
    4210, 4409, 4608, 4807, 5006,
    5205, 5403, 5601, 5799, 5997,
    6195, 6392, 6589, 6786, 6982,
    7179, 7375, 7571, 7766, 7961,
    8156, 8351, 8545, 8739, 8932,
    9126, 9319, 9511, 9703, 9895,
   10087, 10278, 10469, 10659, 10849,
   11038, 11227, 11416, 11604, 11792,
   11980, 12166, 12353, 12539, 12724,
   12909, 13094, 13278, 13462, 13645,
   13827, 14009, 14191, 14372, 14552,
   14732, 14911, 15090, 15268, 15446,
   15623, 15799, 15975, 16150, 16325,
   16499, 16672, 16845, 17017, 17189,
   17360, 17530, 17699, 17868, 18036,
```

```
      18784, 18871, 18937, 18782, 18867,
      19031, 19194, 19357, 19519, 19680,
      19840, 20000, 20159, 20317, 20474,
      20631, 20787, 20942, 21096, 21249,
      21402, 21554, 21705, 21855, 22004,
      22153, 22301, 22448, 22594, 22739,
      22883, 23027, 23169, 23311, 23452,
      23592, 23731, 23869, 24006, 24143,
      24278, 24413, 24546, 24679, 24811,
      24942, 25072, 25201, 25329, 25456,
      25582, 25707, 25831, 25954, 26077,
      26198, 26318, 26437, 26556, 26673,
      26789, 26905, 27019, 27132, 27244,
      27355, 27466, 27575, 27683, 27790,
      27896, 28001, 28105, 28208, 28309,
      28410, 28510, 28608, 28706, 28802,
      28897, 28992, 29085, 29177, 29268,
      29358, 29446, 29534, 29621, 29706,
      29790, 29873, 29955, 30036, 30116,
      30195, 30272, 30349, 30424, 30498,
      30571, 30643, 30713, 30783, 30851,
      30918, 30984, 31049, 31113, 31175,
      31236, 31297, 31356, 31413, 31470,
      31525, 31580, 31633, 31684, 31735,
      31785, 31833, 31880, 31926, 31970,
      32014, 32056, 32097, 32137, 32176,
      32213, 32249, 32284, 32318, 32350,
      32382, 32412, 32441, 32468, 32495,
      32520, 32544, 32567, 32588, 32609,
      32628, 32646, 32662, 32678, 32692,
      32705, 32717, 32727, 32736, 32744,
      32751, 32757, 32761, 32764, 32766,
      32767];

    INTEGER P2 [1,2,4,8,16,32,64,128,256,512,1024,2048];

    INCLUDE TRUFALSE.DEF.I;
    INCLUDE IOSUBS.LIB.I;
    INCLUDE REALCON.LIB.I;
    INCLUDE REALIOC.LIB.I;
    INCLUDE FMUL.LIB.I;

PROCEDURE SIN_FOLD(INTEGER INDEX);
   IF INDEX AND 256 THEN RETURN TABLE(N2-INDEX);
ENDPROC TABLE(INDEX);

PROCEDURE COS_FOLD(INTEGER INDEX);
   IF INDEX AND 256 THEN RETURN  TABLE(INDEX-N4) OR $8000;
ENDPROC TABLE(N4-INDEX);


PROCEDURE INPUT_FUNCTION : INTEGER I,SUM;
   I :=0; SUM = -8192
   WHILE I <N
   BEGIN
      XIRE(I) = SUM
      XIIM(I) = 0;
      SUM = SUM+256;
      IF SUM > 8188 THEN SUM = -8192;
      I = I+1;
   END;
ENDPROC;

PROCEDURE FASTFOUR;
   INTEGER A,B,C,D,E,F,J,I,A,IN2J,POWER2,NDP2,P2D2,SINW,COSW,ARE,AIM;
   BYTE J_ODD;

   J = 1;
   WHILE J <=NSTAGE
   BEGIN
      IF J AND 1 THEN J_ODD = TRUE ELSE J_ODD = FALSE;
      POWER2 = P2(J);
      NDP2 = N / POWER2;
      P2D2 = POWER2 / 2;
      I=0;
      WHILE I < P2D2
      BEGIN
         IN2J = I*NDP2;
         COSW = COS_FOLD(IN2J);
         SINW = SIN_FOLD(IN2J);
         F = SHIFT(IN2J,1);
         A = I;
         WHILE A < NDP2


         BEGIN
            A = A+IN2J;
            B = A+F;
            C = B+NDP2;
            E = A + N2;
            IF J_ODD THEN
            BEGIN
               ARE = FMUL(XIRE(C),COSW) + FMUL(XIIM(C),SINW);
               AIM = FMUL(XIIM(C),COSW) - FMUL(XIRE(C),SINW);
               X2RE(A) = SHIFT(XIRE(B) + ARE,-1);
               X2IM(A) = SHIFT(XIIM(B) + AIM,-1);
               X2RE(E) = SHIFT(XIRE(B) - ARE,-1);
               X2IM(E) = SHIFT(XIIM(B) - AIM,-1);
            END
            ELSE
            BEGIN
               ARE = FMUL(X2RE(C),COSW) + FMUL(X2IM(C),SINW);
               AIM = FMUL(X2IM(C),COSW) - FMUL(X2RE(C),SINW);
               XIRE(A) = SHIFT(X2RE(B) + ARE,-1);
               XIIM(A) = SHIFT(X2IM(B) + AIM,-1);
               XIRE(E) = SHIFT(X2RE(B) - ARE,-1);
               XIIM(E) = SHIFT(X2IM(B) - AIM,-1);
            END;
            A = A+1;
         END;
         I=I+1;
      END;
      J = J+1;
   END;
ENDPROC;


PROCEDURE MAG(REAL X,Y): REAL RESULT;
   RESULT = SQR(X*X+Y*Y);
ENDPROC REAL RESULT;


PROCEDURE OUTPUT: INTEGER I : REAL VALUE, VALUE1;
   VALUE1 = MAG(XIRE(16), XIIM(16));
   I=0;
   WHILE I <N2
   BEGIN
      VALUE = MAG(XIRE(I),XIIM(I));
      PRINT(I,3);
      PRINTF(VALUE/VALUE1,12,6);
      PRINTF(VALUE,16,6);
      CRLF;
      I=I+1;
      IF I\22 =0 THEN GETCHAR;
   END;
ENDPROC;


PROCEDURE MAIN;
   INPUT_FUNCTION;
   CRLF;
   PRINT("START PROGRAM");
   CRLF;
   FASTFOUR;
   CRLF;
   OUTPUT;
```

Basically OS-9™

By: Ron Voigts
2024 Baldwin Court
Glendale, Il 60139

YOU GOTTA HAVE A SYSTEM

One topic I have been wanting to talk about for some time has eluded me. Somehow the column has turned to other things. Perhaps it should have been one of the first things this column discussed. It is the OS-9 System and creating bootable disks.

Computer systems are the one thing that most people don't think about. They're taken for granted. It is assumed that they'll be there. And for the most part, the important thing is to be able to interact with them. Who cares about what is there and how it got there?

For someone working on a large, multiple user system everything is always there. The individual does his own thing. The system is ready to work when he is ready. On smaller systems, like the home computer, everything is on ROM so the systems comes up when the power is turned on. Or maybe with the newer system, you put in a disk and everything comes up from the disk automatically. No thought is involved. Just plod ahead and use it.

The first computer system I used was an 8 bit one that ran on a S-100 bus. This was back in the days when I was first learning how to program Basic and only had a small incling of what an assembler was used for. I wanted to create a disk that had my personal programs on it. I copied the basic programs from a bootable disk to mine and added my own programs to it. The net result was a disk that looked like the original, but it wouldn't boot. An associate, I worked with, used the same type of system. He informed me that my disk did not have a 'system' on it. The operating system resided on track 0 and it had to be copied over from a bootable disk with a special copying program. My personalized, bootable disk was saved.

Later we used a 16 bit system running on a CAMAC environment. By this time, I and my associates understood the worth of a system. But we found that our system was inadequate for what we intended to do. We needed multi-terminal support, but were only capable of interacting with one terminal. I made a quick trip to one of our computer geniuses (self proclaimed, of course!). He invoked a program called SYSGEN. It asked him numerous questions regarding internal and external hardware, user demands, and other necessary information. When it had finished, we had a new 'system' that was capable of interacting with more than one terminal. The morale? *YOU GOTTA HAVE A SYSTEM!*

THE OS-9 SYSTEM

Without a set of instructions the computer is totally useless. It has to have something to do. The trick is to get the information into it. The easiest solution and most widely used it to put in a ROM with some instructions. Not a lot is needed. Just enough information to get the whole thing going. With a minimal amount of instructions the computer can pull itself up by its own bootstraps and load more necessary information from some external media. Hence, the term "booting" a computer is used to refer to the process of starting up the system.

On OS-9 Level 1, the ROM would contain INIT, BOOT, OS9P1 and OS9P2. OS9P1 and OS9P2 make up the Kernel. The kernel provides system services, like multi-tasking and memory management. INIT is the initialization table used by

the kernel at startup. BOOT loads all modules into memory at startup. These modules are essential for system startup.

Level 2 differs slightly. The OS9P2 modules is loaded at startup instead of being ROM resident. The Color Computer does not have any of these modules in ROM. Rather it has them located on track 34 of a bootable disk. This track is loaded into memory at startup. This also explains why when doing a DCHECK on a bootable Color Computer disk, a number of sectors will be found to be allocated, but not in the file structure. These are the ones on track 34.

To get things going, some affirmative action must be taken. It may be only turning the computer on for an auto starting system. Or some system require entering a character from the keyboard like the letter "O". Color Computer users type "DOS" or run a boot program from RSDOS. Whatever, things start happening. The kernel takes over. It initializes the system. A search is made for ROM's in memory. The amount of RAM is determined. OS9Boot is loaded from the bootstrap device. And the system startup task, SYSGO, is run. Within seconds, the familiar prompt of the SHELL is displayed and everything is ready to go.

The OS9Boot file contains the balance of the modules that are part of the system. Standard ones, found in the file are IOMan, RBFMan, SCFMan, and PipeMan. Other modules types are device drivers and descriptors. These handle the basic I/O funtions with things like terminals, modems, disk drives and printers. Optionally, other modules can be added to the 'boot' file. Many users like to have basic commands part of the system. So they add ones like DIR, LIST, and DEL. The advantage is the module is always ready to be used and there is a memory savings, since it gets allocated in a continuous block of memory, instead of at the start of a memory page. The disadvantage is that modules cannot be unlinked from memory. The only way to get rid of them is to start over with a system disk that doesn't have the unwanted modules. I prefer to use load and link instructions in my STARTUP file. Later, I can unlink something I don't.

In Logical Sector Number 0 on the bootable disk are two locations that tell the location and size of the OS9Boot file on the system disk. They are DD.BT and DD.BSZ. DD.BT is a 3 byte number located at relative address $15. This is the starting sector of the 'boot' file on the disk. DD.BSZ is a 2 byte number at $18. It shows the size of the file. There are no provisions for fragmenting the file. The OS9Boot must be in a continuous area on the disk.

MAKING A SYSTEM DISK

The simplest method of making a bootable disk is to use BACKUP and create a clone of the original disk. Then unwanted files can be deleted and new ones added. This method does not allow you to customize a disk. Also, the disk get fragmented from removing modules. Larger files added to the disk will be located at various locations. This will result in slower access time.

Another method is using COBBLER. As we all recall, a cobbler is a man who makes boots. Good name for something that creates a boot disk! Cobbler generates a new system on a device specified in the command line. Entering:

```
OS9:COBBLER /D1
```

will create a bootable disk on drive 1. Again, the disk must contain enough continuous memory to store the OS9Boot file. Also, for Color Computer OS-9, it writes the kernel to track 34. If something is already there, a warning will be displayed.

```
WARNING - FILE(S) OR KERNEL
PRESENT ON TRACK 34 - THIS
TRACK NOT REWRITTEN
```

It is best to start with a freshly formatted disk.

Ultimately, OS9GEN is best for creating a system disk. Level 2 users have no choice, since they can't use Cobbler. And OS9GEN can be used to create custom disk. The OS9Boot can be changed as desired. To add a new driver to the system, we enter:

```
OS9:OS9GEN /D1
/D0/OS9Boot
/D0/DRIVERS/ramdisk
<ESCAPE>
```

The OSGEN will get the modules from /D0 and create a new boot disk on /D1. 'Escape' is entered on the last line to terminate. Color Computer owners use <CLEAR><BREAK> to generate an escape.

Another method is to create a directory on a freshly formatted disk. In the new directory, place modules you want in the new system. Use SAVE to copy them from memory. Ones like IOMan, SCF, RBF and PipeMan are done this way. You will also want drivers and device descriptors. These are either taken from memory or can be copied into the directory. Next create a file with the module names, putting a module per line. The modules found in ROM or on track 34 for Color Computer OS-9 are not placed in the list. Let's say we have a directory on D1 called MODULES and a list file, appropriately called 'modlist'. These do not take up an immense amount of room so we can create the system on the same disk. Enter:

```
OS9:chd /d1/modules
OS9:os9gen /d1 </d1/modlist
```

It will read the list from 'modlist' and use the modules in the directory. When finished there will be a bootable disk in /D1. Use DEL to eliminate 'modlist' and DELDIR for the module directory. Now copy what you want to the new bootable disk.

Some bits & bites

In several sections of this issure you will note that references are made to our Data-Comp's MUSTANG-020™. That has it seems upset a few (actually 2) readers. However, *because I have always felt that even I complaint was sufficient to reply to,* I will drop in here a short reply. Everyone of you deserves an answer.

First, because Data-Comp is a part of the CPI family, we would naturally feel inclined to mention it. Secondly..thirdly..etc..we are the source of all information, as concerns that particular product. We, and only we, can bring you timely information concerning it. And if you will notice, the majority of our mentions of the MUSTANG-020, in this and past issues, is the same kind of information we publish for any other product. We tell you what it is, where to buy it, what the price is, what is changing or in its future, etc.

The only problem I might see is that we have access to more actual information concerning the MUSTANG-020 than practically any other system or product advertised in 68 MICRO JOURNAL (it is advertised). If other advertisers (and many who do not advertise) would send us as much information on their products, as we have on the MUSTANG-020, I guess someone would complain that we were talking too much about that product (it has happened)! Or it just might be a case of - *sour grapes!* So, we do talk about the system, *as it is one of the fastest selling system advertised in 68 MICRO JOURNAL!*

Now, we didn't cause that (we do like it though), actually it is because we were willing to take the product development, advertising and other unknown cost risk, that some of the others were not willing to take. Actually, with us and our other sales efforts, that has always mostly been the case! *But regardless of the reasons no one else did it...we did, and it is working.*

As to the other mentions of the MUSTANG-020 in this and others issues of 68 MICRO JOURNAL, they come from outside sources. In every instances where the MUSTANG-020 was mentioned in these pages, it was done without any request on our part. The only time we have every meddled with what was written by someone else about the system was to correct its spelling, or some other trivial editorial matter. **We never asked to be mentioned, but certainly appreciate it that we were!** However, it was the system that prompted the mention, not myself, or anyone on our staff!

What is important is that we gave them in their purchase of the MUSTANG-020, no special price. They bought it the same as any of you could. They mentioned it (even raved sorta) because it is just that good!

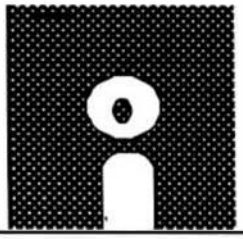
Also, I might mention that there will other MUSTANG-XXX system coming. Some will be in price ranges that will better suit many of you. But first you ought to check out the limited time price in the ad on page five of this issue. Oops...I did it again.

DMW

- - -

C User Notes™

By: E. M. (Bud) Pass Ph.D
Computer Systems Consultants
1454 Latta Lane, N.W.
Conyers, Ga 30207
404 483-1717/4570



INTRODUCTION

This chapter continues the discussion of C compilers (for the 68000 family) and describes some of the concepts behind the character-handling functions of a text editor.

68000 SYSTEMS

I recently used UNIFLEX/68000 on a Mustang-020 for some file and program conversions and I currently use OS-9/68000 on a VME-10, so I have first-hand experience with both sets of operating systems, computers, utility sets, and C compilers. Each set has its own strengths and weaknesses, which are discussed below.

LOSS OF CARRIER

I encountered problems in using both machines remotely. I could find no manner in which to make either sensitive to loss of carrier on their modems, at least without modification of the device drivers. This causes several problems.

One problem involves security. If a user logs in remotely, but does not or can not log off, the port is already logged in when the next user dials in. The next user could delete or modify all or some of the other user's files, list sensitive information, etc. This problem is only partially solved by requesting that each user log off explicitly before dropping the line or dial back in and then log off.

Another problem involves access. If a user logs in remotely and subsequently places the port into raw mode in preparation for full-screen editing, computer-to-computer data transmission, etc., but then does not or can not return the port to normal mode, the port will be unusable until reset locally or through another remote access. If the system is unattended and there is only one modem attached, the system will be unavailable for remote access until someone physically corrects the situation, perhaps by rebooting the computer.

On systems sensitive to changes in carrier detect, users may be logged out and their ports may be reset to a normal state upon loss of carrier detect, thus correcting both of these problems. Hopefully, this is a software oversight in both systems which will be corrected in a timely manner.

68000 C COMPILER FAMILY TIES

Both of these 68000 C compilers were obviously derived from the 6809 McCosh family of C compilers, as they share certain recognizable characteristics. They are almost full C compilers, lacking only bit fields and other minor features. The UNIFLEX/68000 C compiler implements structure assignments and enum and void data types, which the OS-9/68000 C compiler does not implement.

In both compilers, structures containing members which are character strings instead contain pointers to the character strings, not the strings themselves. This works correctly in most situations, but is confusing to someone trying to learn the C language from one of several texts which emphasize the difference between members which are pointers to strings and members which are character strings. Thus, the following structures share the same sizeof(struct) and organization in memory in this family of compilers:

```
struct x1 {char c[10]; };

  and

struct x2 {char *c; };
```

Also, both compilers and linkers are quite restrictive on the use of extern variables and the linkage of variables across modules. The UNIX V and many other C compilers are significantly less restrictive in their handling of variable linkage. The UNIX V C compiler does not normally require the use of extern, automatically linking variables and functions by name across modules, and giving error messages only in case of missing names or definition conflicts.

With all of these restrictions, some programs are difficult to ever get to link properly. Thus, the following example program does not link correctly on current versions of this family of compilers.

```
/* header.h file for global definitions  ********** */

UNIV char comstr[100];
UNIV char secnd[50];
UNIV char third[100];


/* module1.c                          ********** */

#include <stdio.h>
#define UNIV
#include "header.h"

main()
{
  puts("in module 1 \n");
   strcpy(third,"this is the third array\n");
  module2();
  puts("we have finished\n");
}

 /* module2.c                         ********** */

 #include <stdio.h>
 #define UNIV extern
 #include "header.h"

 module2()
 {
    puts("we are now in module 2\n");
  strcpy(comstr,"this is a test message\n");
  module3();
}


/* module3.c                          ********** */

#include <stdio.h>
#define UNIV extern
#include "header.h"

module3()
{
   puts("we are now in module 3 \n");
   strcpy(secnd,"this is a second value\n");
    puts(third);
   puts(secnd);
    puts(comstr);
}


/* correct output (from UNIX V)       ********** */

in module 1

we are now in module 2

we are now in module 3

this is the third array

this is a second value

this is a test message

we have finished
```

68000 C COMPILER AND SYSTEM PROBLEMS

The problems that I encountered with the UNIFLEX/68000 C compiler were reasonably minor ones primarily related to the printf family of functions.

The UNIFLEX/68000 C compiler does not recognize the pflinit function, although it does recognize the pffinit function. This is a nuisance for someone concerned with program portability, as the pflinit (pffinit) function is required on the 6809 version of the McCosh C compilers to cause the inclusion of the version of the printf family of functions which correctly process long (float) data types. The OS-9/68000 C compiler recognizes both functions, although it treats them as dummy calls with no useful effect.

The UNIFLEX/68000 C compiler does not process the following statement correctly:

```
printf ("%05d", 5);
```

in that it omits the leading zeroes specified in the format string.

The OS-9/68000 C compiler has been updated (to version 2.0), and Microware sent a list of the changes, some of which are listed below.

The compiler has two new options and checks two environment variables which specify the include and linker search libraries.

The compiler supports names of up to 256 characters in length.

The preprocessor supports the names __LINE__ and __FILE__ to indicate the current source line number and file name.

The compiler allows non-unique structure and union names, but requires that member names be associated with the designated structure or union.

The library contains several new functions and includes to allow access to additional system functions.

The problems that I encountered with the OS-9/68000 C compiler (version 2.0) were also reasonably minor, but are potentially bothersome.

SOMETHING FOR ALL OF US

FROM ALL OF US

The first phase of the OS-9/68000 C compiler accepts a variable named pc, but the assembler phase rejects it, since pc is the name of a 68000 register. The compiler should check for variable names reserved by the 68000 assembler and change them to prevent this situation.

The OS-9/68000 C compiler bombs (with a bus error) while processing an initialized float array, in a statement such as the following:

```
float interp[10] =
   {0, 0.5, 0.3333, 0.25, 0.6667,
    0.2, 0.5, 0.4, 0.75, 0.6, 0.8};
```

It does not bomb if the initialization is deleted.

The OS-9/68000 C compiler produces several warning messages which cannot apparently be suppressed and often obscure more important messages. Their generation can always be stopped on an individual basis, but this is a nuisance in large programs being ported to the OS-9/68000 system. The warnings most often produced are as follows:

Warning - Possible degenerate assignment

{in statements like: if (x = (d == ':'))}

Warning - Label unused

The default destination of the object file produced by the OS-9/68000 C compiler is the execution directory. If the object file should not be placed in the system cmds directory, the user must remember either to set the execution directory before starting the compilation or must provide an explicit path on the command line for the object file.

Several bad actions can occur. The test version of a program can easily overlay the production version of the same program. If the user does not have permission to overwrite the program in the execution directory, the compiler will issue an error message and discard the new object file.

Even if the user manages to place the object file in the current data directory, OS-9/68000 will not execute it as an object file unless its full path is provided or the execution directory is set to the same file as the data directory. Since OS-9/68000 attempts to execute the object file as a script file, the first two bytes of an object file are hex "4AFC", and hex "4A" is another representation of an ASCII "J", the error message is always "Could not open J", which is obscure and confusing to most users. OS-9/68000 should check the first two characters in a file about to be executed in order to determine if it is a text file or an object file, regardless of whether it is in the data directory or in the execution directory.

Another minor irritation is that the linker assumes a minimum stack size (2K) (Editor's NOTE: AMEN! - DMW) for the object program, so that the user must remember to override the default allocation at link time or to override the allocation on each execution of the program.

TEXT EDITOR

In a previous chapter, a one-window screen editor written using CURSES functions was presented. If CURSES is available on a given system, the internal edit problems are solved. If CURSES is not available on a given system, the program can still be used as a model for the development of similar capabilities. Also, to make the editor more practical, it must be expanded to handle more than one screen. The following discussion is intended to cover both of these topics. Some of the character string manipulation functions appear in the example section below.

Although it occupies at least 1920 characters (for a 24 by 80 screen), many text editors use a screen-image buffer to hold the current window being edited. This simplifies many of the character-handling functions dramatically when compared with the alternative of editing data in the original data structures and also assists in the optimization of screen updating. For the purpose of the discussion below, the use of such a buffer is assumed.

This window is a virtual image to the user of the data in the file being edited. The screen display is normally almost independent of the data structure being used, with the exception that in many editors which handle files larger than main memory, the user is required to explicitly page the memory window thru the file.

When a window is being constructed, the selected text is moved from the structure holding it into the screen buffer. All character manipulation is done within the screen buffer. When lines are inserted or deleted and before another window is constructed, the structure is updated from the screen buffer.

The primary operations on the screen buffer in many editors are as follows, with the corresponding curses functions:

operation	curses
update buffer from structure	

```
get characters from keyboard   getch
update screen                  refresh
move cursor within buffer      move
modify characters in buffer
   insert                      insch
   delete                      delch
   change                      addch
modify lines in buffer
   insert                      insertln
   delete                      deleteln
modify words in buffer
   insert
   delete
   change
   next
   previous
update structure from buffer
```

Curses attempts to optimize the screen updates by performing them only when the program calls the refresh function. In the limited application to a text editor, this is extremely inefficient, as the screen must be refreshed on most keystrokes, and effective screen update optimization can be done by comparison of data being placed into the screen buffer with the data already present there, on a character by character basis.

The word operations listed above are composite character operations based on the location of word delimiters. Most modern editors support word-oriented and even more sophisticated operations, such as undo last operation and automatic line-length adjustment and pagination.

C PROBLEM

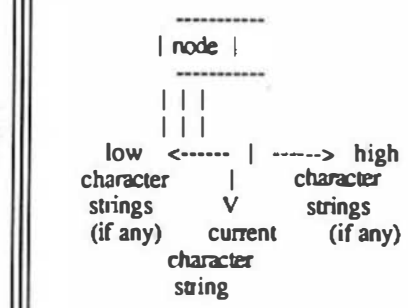
The previous problem was to write a function to dump the b-tree in alphabetical order. The following structure definition and function definitions and calls indicate possible solutions to this problem.

```
/*
 * b-tree structure
 */
struct  btree
{
   char *str;          /* character string pointer */
   struct btree *lo, *hi;  /*  low and high pointers   */
}
   *root = NULL;

/*
 * dump b-tree in alphabetical order
 *  (recursive-only version)
 */
dumpbtree(node)
struct btree *node;
{
     if(node)
     {
          dumpbtree(node->lo);
          puts(node->str);
          dumpbtree(node->hi);
     }
}
 /*
  * dump b-tree in alphabetical order
  *  (iterative-recursive version)  (first choice)
  */
 dumpbtree(node)
 struct btree *node;
 {
      while (node)
      {
           dumpbtree(node->lo);
           puts(node->str);
           node = node->hi;
      }
 }

 /*
  * call to dump b-tree
  */
      dumpbtree(root);
```

A b-tree node may be symbolically represented as follows:



Both of the functions listed above recurse on the lower strings until the lowest string is found. They print this string, then the next lower, etc., alternately unwinding the recursion and recursing or iterating on the higher strings, until the entire list is printed.

An iterative-only version of the dumpbtree functions above is certainly possible, but is substantially more complex than either of the functions shown above, because a table of unknown size is required to keep track of where in the b-tree the function is currently operating. The size of the table is determined by the maximum depth of the b-tree.

SOMETHING FOR ALL OF US

FROM ALL OF US

The expected depth of the b-tree may be readily computed, assuming a random initial sequence of strings from which it is loaded and no further modifications. It is proportional to the log (to the base 2) of the number of entries in the b-tree. This is for the case of a b-tree with one string and two links per node, like the one described above.

In many cases, b-trees can provide an efficient replacement for sorting. If it is necessary to store duplicate strings in the b-tree, either the duplicate strings could be stored for later retrieval or a duplicate count could be added to each node, either with only minor revisions to the loading algorithm.

B-trees may exhibit pathological situations if they are loaded or modified in non-random manners. If a b-tree is loaded from an ascending sequence, all of the lower string pointers will be null. If a b-tree is loaded from a descending sequence, all of the higher string pointers will be null. In both cases, the other pointer will form a sequential linked list through all of the loaded character strings.

Searching either of these sequential b-trees is very slow and inefficient, even slower than even a sequential table search, because of the necessity of following the links. However, attempting to apply the first function listed above to either case or attempting to apply either function to the second case will fail due to stack overflow if the b-tree contains a sufficiently large number of strings. An iterative-only function would be hardly better in these pathological cases, since the number of entries in the table could be the same as the number of items in the b-tree.

If a b-tree is loaded from a random sequence, but is then modified in non-random manners (such as deleting every item in a certain key range or adding a large number of items in a close range), parts of it may become badly un-balanced.

The concept of the balanced b-tree is designed to to overcome the problems involved in processing unbalanced b-trees. For the next C problem, consider ways in which operations on b-trees may be defined to attempt to keep them better-balanced than the those produced by the basic b-tree processing functions. Of course, this is an open-ended problem, probably with no definitive answer.

EXAMPLE C PROGRAM

Following is this month's example C program; it is composed of several functions which modify characters in a screen buffer.

```
#include <stdio.h>

int rows = 24; /* lines per screen */
int cols = 80; /* columns per row */

int putat(); /* output character ch at (y,x) */

addch(ch)
unsigned int ch;
{
 int x = stdscr->_curx, y = stdscr->_cury;

 if (ch != stdscr->_y[y][x])
 {
  putat(y,x,stdscr->_y[y][x] = ch);
  stdscr->_curx = x + (x < cols);
 }
}

delch()
{
 int x = stdscr->_curx, y = stdscr->_cury;
 unsigned int ch;

 for ( ; x < cols; ++x)
 {
  ch = (x < (cols - 1)) ? stdscr->_y[y][x + 1] : 0x20;
  if (ch != stdscr->_y[y][x])
   putat(y,x,stdscr->_y[y][x] = ch);
 }
}

insch(c)
unsigned int c;
{
 int x = stdscr->_curx, y = stdscr->_cury, z;
 unsigned int ch;

 for (z = cols - 1; z >= x; --z)
 {
  ch = (z > x) ? stdscr->_y[y][z - 1] : c;
  if (ch != stdscr->_y[y][z])
   putat(y,z,stdscr->_y[y][z] = ch);
 }
}

main()
{
 /* dummy */
}
```

"HIER & OTHER THINGS"

By: Mickey Ferguson
POB 87
Kingston Springs, TN 37082

FOR THOSE WHO NEED TO KNOW

If you are a regular reader of *68 Micro Journal* (aren't we all?), you might remember an article I wrote for the Feb 86 issue of *68 MICRO JOURNAL* about adding hard disk drives to my 6809 system. In that article I bemoaned my inability to boot from the hard disk and, more importantly, the lack of sub-directories in FLEX. Well, about the time the article was published, I solved the problem of booting from the hard disk. If you are interested in how, please send me an S.A.S.E. and ye shall know.

If you intend to use the method that I used to boot from the *WellWritten* hard disk system; you MUST have their disk drivers in ROM and also be able to either re-assemble or patch the version of FLEX that you are using. But more important, I was inundated with letters (and floppy disks) from people who had added a sub-directory system to FLEX. All of these were different from each other, but (except for HIER) all shared one thing in common which is they divided the hard disk into segments and called each of these a directory. It really astonished me to see how many people had attacked this problem and solved it in their own way. But reached the same basic solution as most everyone else.

HIER is different. It is a true hierarchical file system and any directory may use any amount of available disk space. HIER is a patch to FLEX and a collection of utilities. Some of the utilities are replacements for existing FLEX utilities while others are new utilities made necessary by the sub-directory system. When you receive HIER, you get two disks. One of these contains only one program called INSTALL. This program checks thru the version of FLEX it finds in memory and attempts to tell you what patches must be made to use HIER.

If you do not feel that you can successfully install HIER, you are advised to return the package for a refund (assuming that you have not opened the the disk that contains HIER and the utilities). I cannot comment as to the effectiveness of the INSTALL program except to say that it told me that no changes were necessary, and none were. However, I did modify HIER because it is yet another relocatable program that moves itself to the top of memory and adjusts the MEMEND pointer to protect itself from being overwritten. Since Ray Goff includes the source to HIER, I reworked it to be non-relocatable and as compact as possible to fit in the small EPROM space that I had available. This meant that I had to modify FLEX to know that HIER was always there in EPROM when FLEX was booted.

"THE UTILITIES"

The utilities included with HIER are as follows: *BACKUP, CHGDIR, COPY, DELDIR, DIR, DISCCHK, HOME, LIST, MAKEDIR, MOVE, RUN, SETDIR, TREE, UNDER, and WHERE*. A description of these utilities follows.

BACKUP allows you to copy

all files in the current directory and all files in all sub-directories. This is extremely useful if you have two hard disks and wish to copy an entire disk to another including all files in all sub-directories. If you have only one hard disk, BACKUP makes it very easy to copy a sub-directory to a floppy disk.

CHDIR is used to change to a different directory; with it you can move the current directory in three different ways; to a higher level directory, to a lower level directory, or to a different directory on the same level.

COPY works just like the COPY.CMD included with FLEX except that a directory path name may be included in the SOURCE file spec. Files are always copied into the current directory in the destination file spec, thus path names are not allowed in the destination. COPY can be used to copy files between directories on the same disk.

DELDIR is used to remove a directory from the disk. Any files in that directory and all sub-directories which it might contain are therefore deleted also. Because this command is about as dangerous as a melt-down; DELDIR trys three times to discourage you from using it!

DIR works like the DIR.CMD utility included with FLEX except that it allows you to specify a path name so that it can give you a listing of the files in any sub-directory on the disk. This is very useful if you are searching for a file. With the regular FLEX DIR command you must first set the current directory and then use DIR to see what files are there.

DISCCHK is used to check the entire structure of the disk. DISCCHK starts in the home directory and continues thru all sub-directories opening, reading, & closing every file. All sector sequence numbers are checked to be sure that file linkages are correct. The sector linkages are kept in a table and the table checked to be sure that two files do not claim ownership of the same sector. Once DISCCHK has finished reading all files and building the table of sectors used; it gives you the opportunity to check the chain of free or un-used sectors on the disk. DISCCHK processes the free sectors in much the same fashion as detailed above and reports the number of free sectors available. It then gives you the opportunity to re-link the chain of free sectors. When you delete a file, FLEX puts the sectors the file used at the END of the chain of free sectors; thus FLEX will only reuse them after all other sectors have been used. Should you choose to have DISCCHK re-link the chain; the chain will start with the first free sector closest to the edge of the disk and will link towards the center of the disk. This can improve the appearant speed of a disk that has had its free chain become broken up by the repeated saving and deleting of files as often happens during program development (or writing articles for 68 Micro!) DISCCHK is as useful on some floppy disks as it is on hard disks.

HOME sets the disk to the root directory. It MUST be used before ANY disk, hard or floppy, can be used after being formatted. HIER stores the location of the current directory on the disk itself and a freshly formatted disk has no current directory assigned. Failure to run HOME will result in a DISK FILE READ ERROR when you attempt to WRITE to the disk, and using DIR will show no files on the disk, even if it is FULL! If someone who does not use HIER gives you something on a floppy disk, you must HOME it before you can use it. When you return the disk, its having been HOMEd will have no affect on their non-HIER system (assuming you did not add any sub-directories to the disk).

LIST works just like the LIST.CMD provided with FLEX except that a directory path name may be used for the file to be LISTed.

MAKEDIR creates a new directory on the disk. It also makes the new directory the current directory on the specified drive. It initially allocates 4 sectors to the new directory, enough room for 40 files. Should you attempt to save more than 40 files in the new directory, FLEX will automatically extend the directory.

MOVE allows you to move a file from one directory to another. The file is NOT copied; instead the directory information is copied from the source directory to the destination directory. It is

then deleted from the source directory. You cannot MOVE a file from one disk to another, only from one directory to another on the SAME disk.

RUN comes in two versions: RUN.CMD & RUN.LOW. RUN allows you to execute a program located in a directory other than the current directory on the specified disk.

SETDIR is used to set the current directory on a given disk. It is much like CHGDIR but SETDIR allows you to specify any directory on the disk.

TREE prints a listing of all directories (and the files in them) on a given disk. TREE starts with the home directory and its files and then proceeds thru all sub-directories and their files.

UNDER is used to determine the number of sectors, files and directories under the current directory on a given disk. This is very useful when using BACKUP to copy from hard disk to floppy. Otherwise it is altogether TOO easy to run out of room on the floppy.

WHERE prints the path to the current directory on a given disk.

As you can see, HIER is a very complete package. Ray tells me that he has been using it for about two years on his system and feels it to be very well debugged. I have been using it extensively for several months and have found no bugs at all. I should warn you that Ray added the MOVE command at my insistance and about MOVE he says: "I have checked MOVE out fairly carefully, but I would not be surprised to hear that there are some problems, since I have only tested it for about half a day." Ray's comment was dated March 28, 1986. As I write this, it is June 4, 1986; and I have not found any problems with MOVE.

Since installing HIER I have created directories for each of the languages that I use, as well as directories for text, etc. HIER has made using hard disks with the FLEX operating system a genuine pleasure and I can't imagine how I ever managed without either the hard disks or HIER! The good folks at WellWritten Software have had the opportunity to use HIER for about as long as I have and they are as impressed with it as I am.

By now you are thinking that I am either 100% sold on HIER or that I have been paid off. Well I do have a few complaints about HIER, some are easily solved while others are not. You see, I am a UNIX person and Ray is a DEC VAX person. This means that HIER uses the VAX operating system syntax when specifying path names and VAX utility names. The utilities can easily be renamed to suit you. So CHGDIR can become CD, for example. But the path names are another matter! HIER uses the following syntax in path names:

[home.asmb.source]flex.txt

If you are more accustomed to using UNIX or a clone, your mind wants to see path names look like this:

\home\asmb\source\flex.txt

(Both examples assume you are specifying the file FLEX.TXT which is in the SOURCE directory; which is a child of the ASMB directory; which is a child of the HOME (or root) directory.)

This might appear to be a very trivial gripe, but if you spend your days at the salt mine working with UNIX (or a clone). When you come home to your FLEX system with HIER, the difference in syntax will have you climbing the walls in short order. If you are not accustomed to working with UNIX (or one of its clones) then this is a very trivial gripe.

To summerize, HIER adds a proper hierarchical file system to the FLEX operating system. It is NOT a kluge! It functions very smoothly and appears to be thoroughly debugged. I have not found ANY bugs in HIER (or any of its utilities) after several months of extensive use. I can not only heartily reccommend it, but will go so far as to say that anyone who is using the FLEX operating system with hard disk drives is in very bad NEED of HIER. I have talked in terms of HIER and hard disks but HIER (and its utilities) function equally well with whatever size floppy disks you might have.

HIER is available from
S.E. MEDIA:
see their catalog - center section.

FLEX is a trademark of T.S.C
UNIX is a trademark of A.T.& T.
DEC & VAX are trademarks of Digital Equipment Corp.

FOR THOSE WHO NEED TO KNOW

READING FLEX DISK SIR *FROM FORTH*

R.D. Lurie
9 Linda Street
Leominster, MA 01453

I think that a lot of the misunderstandings about the *FORTH* language have resulted from illustrating the language with the wrong kind of examples. Usually, the examples are too short to do anything really useful, and they are written to show some particular feature of FORTH, without regard to an audience of beginners or casual readers. With that idea in mind, I would like to present a useful FLEX utility program which illustrates many of the features and conventions of FORTH programming.

The FORTH that I am using runs under FLEX DOS and is the FORTH-83 standard. It was written by W. M. Federici, who has been so gracious as to give it away (see 68'MJ, 2/86)! He calls it "FF9". It is important to keep this in mind, since any disk I/O will be implementation-specific, even though FLEX is the common underlying DOS. Mr. Federici has made use of a jump table and several special variables to make it easy to configure his version of FORTH to various SS-50 systems (mine is a mixture of *SWTP*, *GIMIX*, *Percom*, and home brew).

You will have to modify some of the words to fit a different FORTH you may have, but the program basics would still be the same.

The program listing does not look like the usual FORTH program listing, because it is not divided into the normal "screens" of 16 lines each. Instead, it was written more as a FLEX text file, since, in use, it is stored on the FLEX system disk and loaded from there as needed. This is one of the features of FF9.

Notice that the program is virtually self-documenting when written this way. In many ways, the program listed this way closely resembles a C or Pascal program, if you call each definition a "function" or "procedure", and consider the last definition, .DSKID , to be the main program. After all, FORTH is a highly structured language which won't admit it.

Remember that FORTH programs are not executed in sequence like BASIC. The only way to make this program run it to type the command:

.DSKID

PROGRAM DESCRIPTION

This program is called .DSKID . (Note the punctuation! There is a space between the FORTH word and the period ending the sentence, because there might otherwise be confusion as to whether or not the period is part of the FORTH word.) The first character in the name is . , because this is the usual way to show that displayed output is expected. The . is pronounced "dot" in this context.

The purpose of .DSKID is to read the FLEX "disk identification" which would be important to a FORTH user. Since I have one drive, SSDD80, which I use for my FLEX system drive, and two drives, DSDD80, which I use for FORTH programs and data, it is important to be able to determine which format was used in initializing the disk, as well as the disk name, etc.

The number of free sectors is not displayed, because this number does not reflect the FORTH disk usage. Instead, a separate FORTH directory, manually maintained, is usually stored on the disk. (Yes, it is easy to corrupt a FORTH disk if you are not careful.) Since FORTH programs take the position that "I own the disk", there is less trouble than one might expect.

DUMMY WORD

The definition of TASK is simply a way to mark the beginning of the program. This is done so that the program can be easily erased by commanding:

FORGET TASK

TASK is commonly used in this way in FORTH programs.

CONSTANTS

There is no requirement for doing so, but I like to collect all of my CONSTANTs together at the beginning of the program. The only requirement is that a CONSTANT be defined before it is used. However, I have found that putting them all in one place makes it easier to find them while I am writing a program and later when I am debugging. It also makes it easier to keep the names self-consistant and non-conflicting.

Since all but one of the CONSTANTs referred to RAM in some way, it was easier to use hexadecimal notation. Therefore, the list of CONSTANTs was preceeded by HEX to alert the compiler to its use. After the CONSTANTs were all defined, DECIMAL was used to shift the compiler back again.

The address offset referred to by most of the CONSTANTs is relative to the start of the "screen buffer", which holds four consecutive sectors at one time. FF9 has two such buffers, and it is impossible to know which one would be used by the program at any given time. Therefore, the pointers to RAM must be set dynamically at execution time. This will be explained in more detail later.

SET UP FOR READING TRACK #0

FF9 has a set of disk variables which are used to paramatize disk access; one of these is BASETRK, which sets that track assigned to hold "screen #0". On FLEX double density disks, this is usually track #1, since track #0 is formatted as single density, and its inclusion would foul up the easy conversion of screen numbers to track and sector numbers. However, BASETRK must be set to 0 in order to read the FLEX SIR. Provision for this is made in this section of the program.

Good FORTH programming practice calls for a lot of short definitions, rather than a few long definitions. In the spirit of this philosophy, SET-BASETRK was defined. This word is then used by BASETRK->0 and BASETRK->1 to set the "screen #0" track to zero and one, respectively. Incidentally, notice that FORTH words can be made up of any of the printable ASCII set, and the words can be up to 31 characters long, all significant. Furthermore, the words should be as descriptive as possible.

GET-TRK#0 is the word which does a lot of the work. BLOCK is a standard FORTH word which reads the desired four sectors from the disk and stores the data in the buffer. The address of this buffer is left on the Data Stack, which makes possible the dynamic access to the buffer information.

PROMPT FOR DRIVE NUMBER

It is not worth the effort, nor is it desirable, to write the program in such a way that the desired drive number could be placed on the stack prior to calling for .DSKID . Therefore, it is necessary to prompt for this number.

However, SET-BASEDRV must be defined first. BASEDRV is one of the disk search parameters which can be set by the user. It specifies which drive number, 0, 1, or 2, will be the used as the first drive in the search chain. Since I have three drives and keep my FLEX system disk in drive #0, I normally have BASEDRV set to 1. Therefore, it must be changed to 0 before FORTH can read anything from drive #0. It then must be reset to 1 for normal system operation.

.DSKID-PROMPT comes very close to violating the stand against long definitions, but it is so straight-forward, I decided to let it go. The first thing that happens is that a couple of lines are skipped and the prompt is printed. The system then waits for keyboard input via KEY.

The value for ASCII "0" is subtracted from the input, resulting in a single digit, ranging from 0-9. You simply cannot enter more than one digit, no matter how hard you try! This digit is duplicated, because one copy is lost as a result of the next operation. The digit is compared to "3" (remember, Reverse-Polish notation) and replaced on the Data Stack by a Boolean flag.

The FORTH "IF...ELSE...THEN" construction is confusing, at first, but becomes easy to deal with once the stack function is fully grasped. Essentially, the IF part is processed when the Data Stack holds a TRUE flag, and the ELSE part is processed when the stack holds a FALSE flag. THEN just tells the compiler that this part of the statement is complete. The Boolean flag is removed from the Data Stack as the IF test is made.

If the input number was 0, 1, or 2, it is duplicated and printed; . is the standard FORTH command for printing a number. Execution then jumps down to SET-BASEDRV, with 0, 1, or 2 as the associated number on the Data Stack.

However, if the input number was not 0-2, then check the FLEX work-drive value. If this is selected to be "ALL", then the value read by C@ is $FF. Since the program can't handle this many drives, default to drive #0. On the other hand, if WRKDRV holds 0-2, then use that number with SET-BASEDRV.

The " THEN THEN " is required because of the nested IF statements.

PRINT DISK NAME and EXTENSION

This is the first time we encounter a mysterious floating DUP that just shows up without obvious purpose. Its use is to make a copy of the buffer address and leave it on the data stack. We need to use this buffer address a total of eight times

throughout the program. We could use GET-TRK#0 to return this address each time we need it, but it is much better technique to use GET-TRK#0 only once, and DUPlicate it on the the Data Stack each time we need it.

PRINT-DISK-NAME is a good example of a self-documenting definition. Examination of these three definitions shows that DO-DISK-NAME and DO-DISK-NAME-EXT make it absolutely clear how PRINT-DISK-NAME operates. It is easy to see that the definition of PRINT-DISK-NAME would not be nearly as clear if all three definitions were combined into one.

The pointer to the disk name is calculated by addition of the buffer address and DISK-NAME . The name is printed by TYPE with up to 8 characters. A similar set of operations prints the 3 characters in the extension.

PRINT DISK NUMBER

The disk number is found by the same kind of pointer math used for the disk name and extension. @ is used to fetch the disk number since it is stored as a 16-bit number, but . prints it appropriately, since no leading spaces are printed.

PRINT DISK FORMAT

It was most convenient to split this operation into two words, one dealing with printing letters and the other dealing with printing numbers. DO-SEC has to make a series of choices, selecting between SS and DS and between SD and DD. Since I use only the limited standard list of formatting possibilities, it was easy to say that less than 19 sectors/track means SD, and less than 11 sectors/track means SS. The defaults are DD and DS, respectively.

DO-TRK is merely a matter of reading the "maximum" track number, adding 1, and printing it.

PRINT DISK CREATION DATE

This was, again, simply a matter of reading a single byte of numerical data and printing it. The phrase:

1 .R

controls the printing format. I cheated a little by telling the program to allot only one digit to the number, knowing that two digits would be printed when needed. But, this way, I ensured that there would be no blank spaces between single-digit numbers.

DISPLAY OUTPUT

DISPLAY-DSKID is the word which controls the actual order of printing of the program output. Again, by writing the program in this way, it is self-documenting, and no further comments are necessary.

COMMAND DEFINITION

The program is run by commanding:

.DSKID

The algorithm is evident from the definition of .DSKID:

1. Clear the disk buffers.
2. Get the drive number.
3. Read the Track #0 data.
4. Print the data.
5. Reset BASEDRV.
6. Clear the disk buffers.

FLEX FILE

Of course, .DSKID cannot be used for the name of a FLEX disk file. I chose to use "P-DSKID.FTH", but that has no affect on the FORTH application.

```
( ************************************************************ )
( .DSKID                                          RDL  05/21/86 )
( ************************************************************ )

: TASK ;

   HEX
    30 CONSTANT <0>
   210 CONSTANT DISK-NAME
   218 CONSTANT DISK-NAME-EXT
   21B CONSTANT DISK#
   223 CONSTANT C-MONTH
   224 CONSTANT C-DAY
   225 CONSTANT C-YEAR
   226 CONSTANT TOTAL-TRKS
   227 CONSTANT TOTAL-SECS
  CC0C CONSTANT WRKDRV
   DECIMAL
```

```
( ************************************************************** )
( Set up for reading TRK #0                                      )
( ************************************************************** )

: SET-BASETRK ( --- )          ( Set the number of the first     )
    ['] BASETRK >BODY ! ;      ( accessable disk track           )

: BASETRK->0  ( n --- )
    0 SET-BASETRK ;

: BASETRK->1  ( n --- )
    1 SET-BASETRK ;

: GET-TRK#0   ( --- addr )     ( Load track #0 into one of the   )
    0 BLOCK ;                  ( disk buffers and put its address)
                               ( on the DATA STACK               )

( ************************************************************** )
( Drive prompt                                                   )
( ************************************************************** )

: SET-BASEDRV ( --- )          ( Set the number of the first     )
    ['] BASEDRV >BODY ! ;      ( accessable disk drive           )

: .DSKID-PROMPT    ( --- )
    CR CR ." Drive Number (0-2): "     ( prompt                  )
    KEY <0> -                          ( fetch keyboard input    )
      DUP 3 <                          ( a valid number?         )
        IF DUP .                       ( yes, echo it            )
        ELSE WRKDRV C@ DUP             ( no, fetch FLEX WRKDRV   )
          255 =                        ( does W=A?               )
            IF DROP 0 DUP .            ( yes, so use 0           )
            ELSE DUP . THEN THEN       ( no, so use WRKDRV       )
    SET-BASEDRV ;                      ( select proper drive     )

( ************************************************************** )
( Print disk name                                                )
( ************************************************************** )

: DO-DISK-NAME     ( --- )
    DUP                                ( save buffer address     )
    DISK-NAME + 8 TYPE ;               ( point to name & print   )

: DO-DISK-NAME-EXT ( --- )
    DUP                                ( save buffer address     )
    DISK-NAME-EXT + 3 TYPE ;           ( point to ext. & print   )

: PRINT-DISK-NAME  ( --- )
    ." Disk name: "
    DO-DISK-NAME
    ." ."                              ( print a .               )
    DO-DISK-NAME-EXT ;

( ************************************************************** )
( Print disk number                                              )
( ************************************************************** )

: PRINT-DISK# ( --- )
    DUP                                ( save buffer address     )
    SPACE ." #"
    DISK# + @ . ;                      ( fetch disk# & print     )
```

```
( ************************************************************** )
( Print disk format                                              )
( ************************************************************** )

: DO-SEC ( --- )
    DUP                                  ( save buffer address   )
    TOTAL-SECS + C@
      DUP 19 <
        IF ." SS"                        ( yes, so single side   )
        ELSE ." DS" THEN                 ( no, so double side    )
      11 <
        IF ." SD"                        ( yes, so single density)
        ELSE ." DD" THEN ;               ( no, so dbl. density   )

: DO-TRK ( --- )
    DUP                                  ( save buffer address   )
    TOTAL-TRKS + C@ 1+ . ;

( ************************************************************** )
( Print disk creation date                                       )
( ************************************************************** )

: DO-C-DATE   ( --- )
    DUP DUP                              ( save buffer address   )
    C-MONTH + C@ 1 .R
    ." /"
    C-DAY + C@ 1 .R

      ." /"
      C-YEAR + C@ 1 .R ;

  ( ************************************************************** )
  ( Display output                                                 )
  ( ************************************************************** )

  : DISPLAY-DSKID    ( --- )
      CR CR PRINT-DISK-NAME
      PRINT-DISK#
      DO-SEC DO-TRK
      DO-C-DATE
      CR ;

  ( ************************************************************** )
  ( Command definition                                             )
  ( ************************************************************** )

  : .DSKID ( --- )
      FLUSH                              ( clear old data        )
      .DSKID-PROMPT                      ( fetch drive number    )
      BASETRK->0                         ( enable reading TRK #0 )
      GET-TRK#0                          ( read TRK #0 from disk )
      BASETRK->1                         ( reset to normal cond. )
      DISPLAY-DSKID
      1 SET-BASEDRV                      ( reset to normal cond. )
      EMPTY-BUFFERS                      ( clear old data        )
      CR ;
```

Las Vegas NCC

1986

a bust?

This year may have heralded the end of NCC for the Las Vegas scene. That is the opinion of many who attended the annual affair. Which alternates each year between Las Vegas and Chicago.

I had hoped to return from the *grand* event and report to you about some new and exciting offerings. Well, it just didn't happen. NCC '86, in my opinion, *was a bust.*

The only bright spots that I can report to you on are **GIMIX** and **MICROWARE**, and some new (to me) software and 68XXX hardware. GIMIX and Microware were drawing good crowds.

GIMIX was showing their entire line, including a system configured same as a **MUSTANG-020™**, as well as some new UNIX™ like software (more at a later date). UniFLEX™ from TSC was running on all systems shown. The system they had configured same as a MUSTANG-020 was running to the tune of 20 Mhz. Thats right...20 big 'uns. Now, that is *real Big League stuff!*

Also they were hooked up by Arcnet™ LAN between their booth and the Microware booth. It appeared to be working fine and should be a real plus to all those who have need of such services. More and more we are hearing about networking..of one type or another. Microware should theirs ready for release soon. It's part of the wave of the future in computing.

While the GIMIX 68020 systems were well demonstrated, running everything from BASIC to SCULPTOR+. I saw many who went away wondering about the pricing of 68020 systems from other manufacturers.

SCULPTOR+ was well represented by Bruce Campbell in the GIMIX booth. Bruce is the USA rep for this fine software development system. New Sculptor manuals looked great, some new Sculptor prices, not so great. The market will certainly determine the acceptance of it. However, even at the new rates for some systems, it is still a *bargain* if you are writing any significant programs. Especially those that are of the 'data-base' type. We recommend Sculptor+ as it is well supported and does all it is advertised to do. Bruce is porting it to various other systems. That should make an expanded market for all those developing new software packages.

Microwares booth was a pretty busy place. They had ***OS-9*** running on several systems. One configured same as a MUSTANG-020. Another was the Atari running smoothly with OS-9, level 2. That will be a real nice package when all the necessary support software is available..Volkswagen+ (word processing) and other packages announced at the '86 COMDEX, but not available yet. At least I have not seen any of it. I can't help but believe that when it appears in force (software supported), it will give the CoCo a real run. Actually, I believe it will be *priced less and deliver more* than the CoCo. From that viewpoint I can't believe Tandy will continue the CoCo line much longer.

The CoCo has birthed a cult. A loyal and rather vocal band of supporters, of which we are one. However, users aside, Tandy seems bent on following MS-DOS, whatever its course. Which, in my opinion, spells the end of the CoCo, as we know it.

ASSEMBLERS

ASTRUK09 from S.E. Media – A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
F, CCF - $99.95

Macro Assembler for TSC -- The FLEX STANDARD Assembler.
Special – CCF $35.00; F $50.00

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. GeneOrate OS-9 Memory modules under FLEX.
FLEX, CCF, OS-9 $99.00

Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.
F, CCF $150.00

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.I. Programming for small to medium-sized Programs.
F, CCF - $75.00
XMACE -- MACE w/Cross Assembler for
6800/1/2/3/8 F, CCF - $98.00

TRUE CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/ 146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/ 40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
FLEX, CCF, OS-9, UniFLEX each - $60.00
any 3 - $100.00 - the complete set w/C Source
except the 68000 Source - $200.00
UniFLEX 68000 - $50.00

XASM Cross Assemblers for FLEX from S.E. MEDIA – This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., In providing code for the target CPU's.
Complete set, FLEX only - $150.00

CRASMB from S.E. MEDIA – 8-Bit Macro Cross Assembler with same features as OSM; cross-assemble to 6800/1/2/3/4/5/8/9/11, 6502, 1802, 8048 Sers, 80/85, Z-8, Z-80, TMS-7000 sers. Supports the target chip's standard mnemonics and addressing modes.
FLEX, CCF, OS-9 Full package -- $399.00

CRASMB 16.32 from S.E. MEDIA – Cross Assembler for the 68000.
FLEX, CCF, OS-9 $249.00

UTILITIES

Basic09 XRef from S.E. Media – This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
O & CCOobj. only -- $39.95; w/Source - $79.95

Lucidata PASCAL UTILITIES (Requires LUCIDATA Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER – provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
F, CCF -- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media -- **A UniFLEX BASIC decompiler** Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
U - $219.95

LOW COST PROGRAM KITS from S.E. Media – The following programs are available for FLEX on either 5 or 8 inch disk.

1. **BASIC TOOL CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX UTILITIES KIT $39.95**
   CATS.CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undelates (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: "modularise" disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization where required
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer program
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formated index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F and CCF, U - $25.00, w/Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends–

F = FLEX, **CCF** = Color Computer FLEX
O = OS-9, **CCO** = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No "blind" debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!

Levels I & II only - OS-9 Regular $149.95
SPECIAL INTRODUCTION OFFER *$69.95*

DISK UTILITIES

OS-9 VDisk from S.E. Media – For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.

Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media – Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX Directory, Delete FLEX Files, Copy both directions, etc. FLEX users use the special disk just like any other FLEX disk

O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.

OS-9 $85.00

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX disk (8 - 5 - hard disk) can have sub directories.** By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX like a regular file, except they have the extension ".DIR".** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

"Introduction Special" $69.95

COPYMULT from S.E. Media – Copy LARGE Disks to several smaller disks. FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, 8" or 5") $99.50

COPYCAT from Lucidata – *Pascal NOT required.* Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

F and CCF 5" - $50.00 F 8" - $65.00

FLEX DISK UTILITIES from Computer Systems Consultants – Eight (8) different Assembly Language (w/ Source Code) FLEX Utilities for every FLEX Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or A.L. Source Code).
F and CCF - $50.00
BASIC Utilities ONLY for UniFLEX -- $30.00

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

FLEX, CCF, OS-9, UniFLEX; with complete Source $100.00 without Source $50.00
UniFLEX 68000 with complete Source $100.00

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends–

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

with C-MODEM (58020). Text and binary files may be directly transfered between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks) $99.95
X-TALK Software (2 disks only) $69.95
X-TALK with C/MODEM Source $149.95

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.

U - $299.99

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

*Now supplied as a two disk set:
Disk #1: JUST2.CMD object file, JUST2.TXT PL9 source FLEX - CC
Disk #2: JUSTSC object and source in C: FLEX - OS9 - CC*

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL9 FLEX only - F & CCF - $49.95
Disk Set (2) - F & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

Regular FLEX $129.50
"SPECIAL INTRODUCTION OFFER" $79.95
SPECIAL PAT/JUST COMBO (w/source)
FLEX $99.95
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

Regular $129.95
SPECIAL INTRODUCTION OFFER FLEX $69.95

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.

FLEX, CCF, STAR-DOS Regular $69.95
Limited Special Offer: $39.95

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/STAR-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - $99.95,
For O - $179.95, U - $299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary, Complements Stylograph.

NEW PRICES 6809 CCF and CCO - $69.95,
For O - $99.95, U - $149.95

STYLO.MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - $69.95,
For O - $79.95, U - $129.95

STYLO-PAK -- Graph + Spell + Merge Package Deal!!!

For O - $329.95, U - $549.95
O, 68000 $995.00

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends--

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*OS-9 is a Trademark of Microware and Motorola
*FLEX is a Trademark of Technical Systems Consultants

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including Interrupts. Comprehensive library support; simple Machine Code Interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.
F, CCF - $198.00

PASC from S.E. Media - A Flex9 Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package come complete with source (written in PASC) and documentation.
FLEX $95.00

WHIMSICAL from S.E. MEDIA Now *supports Real Numbers.* "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*
F and CCF - $195.00

FORTH from Stearns Electronics – A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!
Color Computer ONLY - $58.95

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.
CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*
F and CCF - $295.00

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
FLEX, CCF, OS-9 (Level II ONLY), U - $575.00

PASCAL Compiler from Lucidata – ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
F and CCF 5" - $99.95 F 8" - $99.95

PASCAL Compiler from OmegaSoft (now Certified Software) - - For the *PROFESSIONAL*: ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Asmb. and Linking Loader.
F and CCF - $425.00 - One Year Maint. $100.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA – A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.
FLEX, CCF, OS-9 Compiler/Assembler $199.00

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level I COBOL with many of the useful Level 2 features. Full FLEX File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*
FLEX, CCF; Normally $199.00
Special Introductory Price $99.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media – Requires FLEX and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. **This is one of the strongest CHESS programs running on any microcomputer,** *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
F and CCF - $79.95

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends--

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

**** Shipping ****

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler, extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.

Color Computer SS-50 Bus (all w/A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00

DYNAMITE+ -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

CCF, Obj. Only $100.00 - CO, Obj. Only $59.95
F, " " $100.00 - O, object only $150.00
U, " " $300.00

DATA-BASE ACCOUNTING

XDMS from Westchester Applied Business Systems - Powerful DBMS: M.L. program will work on a single sided 5" disk, yet is F-A-S-T. XDMS Level I provides an "entry level" System for defining a Data Base, entering and changing the Data, and producing Reports. XDMS Level II adds the POWERFUL "GENERATE" facility with an English Language Command Structure for manipulating the Data to create new file Structures, Sort, Select, Calculate, etc. XDMS Level III adds special "Utilities" which provide additional ease in setting up a Data Base, such as copying old data into new Data Structures, changing System Parameters, etc.

XDMS System Manual - $24.95
XDMS LVl I - F & CCF - $129.95
XDMS LVl II - F & CCF - $199.95
XDMS LVl III - F & CCF - $269.95

XDMS IV from Westchester Applied Business Systems - *XDMS IV* is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, *but also to process entire files producing customized reports, screen displays and file output.* Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

*XDMS IV - F, CCF STAR-DOS, SK*DOS $350.00*
Upgrades to XDMS IV - $250.00

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants – TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/Source - $100.00

DYNACALC – Electronic Spread Sheet for the 6809 and 68000.

F, OS-9 and SPECIAL CCF - $200.00, U - $395.00

OS-9 68K - $695.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants – Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants - - The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/Source - $100.00

DIET-TRAC Forecaster from S.E. Media – An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.

F - $59.95, U - $89.95

!!! Please Specify Your Operating System & Disk Size !!!

Availability Legends–

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

* OS-9 is a Trademark of Microware and Motorola
* FLEX is a Trademark of Technical Systems Consultants

** Shipping **

Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

A new version (CoCo) will arrive soon, and it should be a dandy. Level 2 OS-9 (Microware would not acknowledge this, *a secret(?)* I suppose), still a 'bit banger' port(s?) and 128 to 512K of RAM. It has a hard battle ahead to ward off the new and coming systems running OS-9. Tandy better rethink the prices (I hear from my trusty sources) or it will configure out way too steep, compared to some of the coming systems. For one, the Atari!

Despite what happens to the CoCo, we will always be indebted to Microware for the excellent support they have given to both the system and it's loyal community of users. It made a lot of folks aware of the *real power* of the 6809. A message we were preaching for years before any of the others even acknowledged that it was *really* around. Again **Thanks** Microware!

Microware was also showing a fine Motorola VME system running OS-9. We hear many good things about the VME bus. And will be doing more in the VME arena, in the months to come. OS-9 is a big force there.

It is apparent that Microware has become a real player in the game of computer operating systems. No doubt it has a lot to do with their attitude towards their users. I hear nothing but good things about their support and cooperation. From underdog right up towards the top, and it looks like clear sailing ahead. What with CD-I and some other things coming.

Ken Kaplan, President and founder of Microware presented a formal forum on CD-I. It was well attended by the press as well as other interested parties. CD-I is the *Wave of the Future* for high fidelity audio, video and data, combined on one media. The operating system is OS-9 and the CPU is a 68XXX device. A combination proven in time and action. We will publish the specification for CD-I in a future issue. I predict that CD-I players, within the next three years, will be *more popular and priced less* than the current VCR market.

I tried to cover all the other vendors that might have something that you would be interested in. Most all the new systems were using the *68020*. The prices varied. A few were trying to dump off 68000 and 68010 systems at prices above most of the 020 systems. However, I did find several 68020 manufactures that had very reasonable prices and were offering good support. Most were UNIX™ systems. I am in the process of gathering additional information, firm prices (some were going to reduce to even lower levels), software available, support levels, vendors, etc. Will let you know in the coming months as I get it all sorted out.

In conclusion, I must admit that I was very disappointed in NCC 1986 - Las Vegas. The crowds were way down. Pre-show estimates were 50-60,000 attendees, 600-900 exhibitors. I counted about 400 or so exhibits. As to attendance...well, at one point I counted badges. Of the 50 or so I noted in one session, 27 were exhibitors, 12 were foreign visitors and the rest were press or paid (assumed). I might add, that if Comdex could get their press registration down as pat as NCC does, it would be a much more enjoyable adventure.

All the cab drivers (who knows better?) were predicting that this just might be the last year for NCC in Las Vegas. Also I caught a real duzzy of a cold on the plane (Delta) going to Vegas...seems their heating system flaked out and we were mostly all wrapped in blankets on arriving in Las Vegas. Sure glad it wasn't the engines or control systems that decided to poop!

So, I guess the market is maturing. I believe shows of this type are on the way out. The print media as well as other means, are doing a better job of educating those who need to know. Also the expense is getting a bit steep. Several I talked to stated that this was their last NCC. Many pulled up stakes and left early. Fact is, some seemed to believe the Comdex shows are a better bet. From my observations of the Spring '86 Atlanta Comdex and the Summer Las Vegas NCC, I agree.

Tell you 'bout Chicago NCC next year?

DMW

- - -

INTERACTIVE KARNAUGH MAPPING

Bradford Taylor
Sharm Engineering
Box 97
Mulvane, KS 67110

I've always hated *Karnaugh mapping*. Never could understand why someone would go to so much trouble just to get a reduced boolean equation. The main problem I had with Karnaugh maps were that simple problems were too trivial to map and complicated problems were too hard to manage with the maps. Of course I was always able to do those darn things for design courses in college, but after the final it was forgotten. Besides, real engineers didn't use Karnaugh maps in the real world. All you needed to know was what a Nand or a Nor gate was and you could remove redundant components by just looking at a schematic.

Of course, it always comes to pass that you must use something that you dislike. The reason in this case was the coming of programmable logic devices such as PAL's, PLA's and the like. Hardware logic design became a software design task rather than a bag of hardware tricks. Also, these devices seemed to work better and were easier to manage if there were fewer terms used in their logic programming.

It just so happened that I was able to attend a CAD seminar on programmable logic devices. During the lecture, all of a sudden, out of nowhere appeared an expensive interactive CAD/CAM color graphics terminal system. One of the software packages that was demonstrated with this system included an interactive Karnaugh map generator that was used to help define logic expression for programmable devices. Well, $10,000 is a hefty amount to pay for a piece of software to say the least, but I was impressed none the less. This was something I had to have for my very own even though I couldn't afford the table the system sat on.

But, I had a few Aces up may sleeve. I knew I had my Heathkit terminal and my trusty FLEX system and of course a 'C' compiler at home. Besides, my time didn't cost $10,000 for a few hours of fun. So, armed with some ideas of my own and my old college text books I set out to develop an interactive Karnaugh map program.

According to the texts, the Karnaugh map is analogous to Venn diagrams of set theory. The products are elements in this theory, and a truth table is equivalent to the set of all elements in this theory. Analogously with set notations it is convenient to denote rows and columns of the Karnaugh map as shown in the following figure:

```
      A
  +---+---+---+---+
  |0|1|0|0|
  +---+---+---+---+
  |1|1|0|1|
  +---+---+---+---+C
  |1|1|0|1|
 D+---+---+---+---+
  |0|1|0|0|
  +---+---+---+---+
      B
```

And, according to the text books, a human being is well trained to perceive two-dimensional figures and to recognize which of them belong together although these areas are scattered over the table. The simplification of Boolean expressions is thus equivalent to setting up a truth table in map form and then trying to find the minimum number of product terms which would cover the 1's in the table. For example, from the figure, the minimum boolean expression is F = ~A*C + ~B*A. (Here, '~' denotes NOT, '*' denotes AND and '+' denotes OR operations.)

It may be easy for a human being to perceive the two dimensional figure, but computers must do things systematically. This became a major challenge. I began implementing the optimization by looking for the groups of two's and four's just like the exercises in the text books. This proved to be a disaster as I was not up to programming image recognition.

Referring back to the text books, I came across the chapter on the Quine-McCluskey method for minimization. For this method, all the true products from the map are arranged in groups depending on the number of bits set in each term. Every row in a group is compared against all the rows in the next group, and whenever a combination is possible, the combined rows are saved into a new table. Combination can occur if terms vary in only one bit and "don't-care" bits must match exactly. This process continues until no more combinations are possible. All duplicate terms are deleted.

Now, having discovered this piece of the puzzle, I was able to complete the task and the result is the program KARN. Adding a truth table display to the screen was frosting on the cake..

The only remaining major task was eliminating the keyboard echo that is associated with the getchar() function for FLEX. I accomplished this by calling the non-echo character input routine directly through the Driver Vector table found at location $D3E5 in FLEX. This proved to be an interesting problem in itself. I could have implemented an assembly language routine to do this, but I wanted to keep the entire program in 'C' for clarity.

This program was compiled and linked using the version *1.6 Introl-C compiler* and linker. Other than the I/O routines, I tend to believe the program can be transported to other systems as I have transported it to one Unix system. Of course the terminal screen-erase and cursor-positioning functions need to be configured to a given terminal. This can probably be handled with the curses package as has been discussed by Bud Pass.

To run the program, just enter KARN and the work screen will be drawn. The J, I, L and comma keys are used to position the cursor left, up, right and down respectively around the Karnaugh map. The 'K' key is used to toggle the elements from 0 to 1. The Escape key is used to return to FLEX. As elements are toggled, the minimized boolean equation is immediatly displayed at the bottom of the screen and the truth table is updated on the right half of the screen.

I have successfully used KARN to help design the logic for a GPIB/RS-232 Spooler and a Audio D/A - A/D signal processing board. Since I was given very little board space for these designs, minimized logic was a requirement. So Karnaugh maps can be fun and profitable.

+++

```
/*******************************************
*
*  Karn -- Generate boolean equations and
*          truth table from interactive
*          Karnaugh map.
*
*  Bradford Taylor
*  Sharm Engineering
*  Box 97
*  Mulvane, Ks 67110
*  Version 1.0
*  May 24, 1986
*
*******************************************/

/*
*** Declare globals
*/
char mat[4][4];    /* Map image      */
char m[48];      /* Minimized truth table */
char dont[48];     /* don't Cares for truth table */
char checked[48]; /* Minimization intermediates  */
char new[48];
char newd[48];
char saved[2][48]; /* Non checked terms     */
char string[256]; /* Expression print string */
int maxterms;      /* Max term counter */
int terms;         /* Number of actual terms */
int row,col;     /* Row and Column counters */
char letters[4] = {'A','B','C','D'};
char order[4][4] = {   /* Actual term order  */
                   0,1,3,2,
                   4,5,7,6,
                   12,13,15,14,
                   8,9,11,10};

#define DELETE 0x50
#define ESC '\033'
#define UP  'i'
#define DOWN ','   /* Cursor control keys   */
#define LEFT 'j'
```

```
#define RIGHT 'l'
#define TOGGLE 'k' /* State toggle key     */
#define forever for(;;)

/*
**** Main entry
*/
main()
{
      erase();     /* Erase screen   */

      for(row=0;row<4;++row)
        for(col=0;col<4;++col)
        mat[row][col] = 0;

       /* Initialize counters  */
       maxterms =
       row      =
       col   = 0;
       grid();          /* draw the grid    */
       help();          /* draw help map    */
       dispchar();       /* display chars    */
       equation();       /* display equation */
       truth();         /* display truth table */
        edit();          /* edit karnaugh    */

}

/*
*** interpret characters and build screen
*/
edit()
{

   forever
    {
     /* position to proper grid    */
     posxy(col,row);

     /* pick proper key action     */
     switch(tolower(getkey()))
     {
      case DOWN: row = (row+1)&3; break;

      case RIGHT: col = (col+1)&3; break;

      case LEFT:  col = (col-1)&3; break;

      case UP:    row = (row-1)&3; break;

     case TOGGLE:
            mat[row][col]= ~mat[row][col];
            square(col,row);
            equation();
            states();
            break;

       case ESC:
            move(22,0);
           return;
       }
     }

}

/*
**** Draw grid
*/
grid()
{
   printat(0,9,"A");
   printat(1,1,"+---+---+---+---+");
   printat(2,1,"|   |   |   |   |");
   printat(3,1,"+---+---+---+---+");
   printat(4,1,"|   |   |   |   |");
   printat(5,1,"+---+---+---+---+C");
   printat(6,1,"|   |   |   |   |");
   printat(7,0,"D+---+---+---+---+");
   printat(8,1,"|   |   |   |   |");
   printat(9,1,"+---+---+---+---+");
    printat(10,13,"B");

}

/*
*** Draw help keys
*/
help()
{
   printat(12,9,"^");
    printat(13,9,"|");
    printat(14,9,"I");
   printat(15,4,"<- J K L ->");
    printat(16,9,",");
    printat(17,9,"|");
    printat(18,9,"V");
   printat(19,4,"DIRECTIONS");
   printat(20,0,"K to Toggle/ESC to Quit");

}
/*
*** Initialize and draw truth table
*/
truth()
{
  printat(0,45,"A  B  C  D  |  F");
  printat(1,45,"-----------+--");
  states();
```

```
}

/*
*** show states of truth table
*/
states()
{
  int numb,i,j,k,bit;

  for(numb=i=0;i<terms;++i)
  {
    for(bit=1,j=0;bit<16;++j,bit+=bit)
    {
     if(dont[i]&bit)          /* Don't Care */
      k = 'X';
     else if(m[i]&bit)    /* High term  */
      k = 'H';
     else
      k = 'L';
     move(numb+2,45+j*3);
     outchar(k);
    }
    printat(numb+2,60,"H");
   ++numb;
  }
  if(!numb)    /* All don't cares? */
  {
   printat(2,45,"X  X  X  X");
    printat(2,60,"H");
   numb = 1;
  }

  /* erase previous states  */
  for(i=numb;i<maxterms;++i)
  {
   printat(i+2,45,"           ");
    printat(i+2,60," ");
  }
  maxterms = numb;
}

hilo(n)
{
      int i;

      for(i=0;i<4;++i)
      {
       if(n&8)
        outchar('H');
       else
        outchar('L');
       n<<=1;
       outchar(' ');
      }

}

/*
*** position character to actual position
*/
posxy(x,y)
int x,y;
{
      move(y*2+2,x*4+3);
}

/*
*** Fill karnaugh map
*/
dispchar()
{
     int i;

     for(i=0;i<4;++i)
      disprow(i);

}

/*
*** display grid row
*/
disprow(r)
int r;
{
      int i,n;

      for(i=n=0;i<4;++i)
      square(i,r);

}

/*
*** fill individual square
*/
square(c,r)
int c,r;
{
      posxy(c,r);
     if(mat[r][c])
      outchar('1');
     else
      outchar('0');
      posxy(c,r);

}
```

```
/*
*** show equation derived from math[]
*/
equation()
{
 char temp[32];
 int  count,l,i,j,bit;


 *string = 0;
 minimize();  /* Quine McCluskey     */
 for(count=i=0;i<terms;++i)
 {
   for(bit=1,l=j=0;bit<16;bit+=bit,++j)
   {
    if(dont[i]&bit)
     continue;
    else if(m[i]&bit)
     ;
    else
     temp[l++] = '~';
    temp[l++] = letters[j];
    temp[l++] = '*';
   }
   temp[l]=0;
   l = strlen(temp)-1;

   /* eliminate trailing '*' */
   if(temp[l] == '*')
    temp[l] = 0;
   strcat(temp,"+");
   if(!strcmp(temp,"+"))  /* all don't cares? */
    temp[l] = 0;
   else
    {
     strcat(string,temp);
     ++count;
    }
 }
 l = strlen(string)-1;
 if(string[l] == '+')     /* eliminate last '+' */
  string[l] = 0;

 if(!count)               /* all full or empty */
   strcpy(string,"1");
  for(l=strlen(string);l<80;++l)
   string[l] = ' ';
  string[l]=0;
  printat(22,0,string);    /* show equation      */

}

/*
*** check if only one bit is set
*/
onebit(b)
int b;
{
return((b&(-b)) == b);
}

/*
*** check if only one bit difference
*** between two bytes
*/
onediff(a,b)
int a,b;
{
 int temp;

 if(onebit(temp = a^b))
  return(temp);          /* return the 1-bit mask */
 return(0);

}

/*
***  Quine - McCluskey Minimization
*/
minimize()
{
  char t,changed;
   int savcnt,ncnt,i,j;


  savcnt = 0;
  /* Collect only the true terms and translate */
  for(terms=i=0;i<4;++i)
   for(j=0;j<4;++j)
    if(mat[i][j])
    {
     dont[terms] = 0;    /*  don't care bits */
     m[terms++] = order[i][j];
    }

  /* Do until no more possible minimization */
  do
  {
   for(i=0;i<terms;++i)
    checked[i] = 0;       /* Checked-off array */
   for(changed=ncnt=i=0;i<terms;++i)
   {
    /*
     **  Compare bits from a bit set group
     **  against members from the next group.
     **  If Don't-Cares match exact and only one
     **  bit is different, add another don't care
     **  and check both members off as merged.
     */
```

```
switch(m[i])
{
 case 0:      /*  No bits group   */
   for(j=0;j<terms;++j)
    switch(m[j])    /* 1-bit group */
    {
     case 1:
     case 2:
     case 4:
     case 8:
       if(dont[i]==dont[j])
        if(t=onediff(m[i],m[j]))
        {
         checked[i] =
         checked[j] =
         changed    =
          newd[ncnt] = dont[j] | t;
          new[ncnt++] = m[j] & ~t;
        }
    }
    break;
 case 1:      /*  1-bit group     */
 case 2:
 case 4:
 case 8:
   for(j=0;j<terms;++j)
    switch(m[j])     /* 2-bit group */
    {
     case 3:
     case 5:
     case 6:
     case 9:
     case 10:
     case 12:
        if(dont[i]==dont[j])
         if(t=onediff(m[i],m[j]))
         {
          checked[i] =
          checked[j] =
          changed    =
           newd[ncnt] = dont[j] | t;
           new[ncnt++] = m[j]&~t;
         }
    }
   break;
 case 3:          /*  2-bit group  */
 case 5:
 case 6:
 case 9:
 case 10:
 case 12:
    for(j=0;j<terms;++j)
     switch(m[j])    /* 3-bit group */
     {
      case 7:
      case 11:
      case 13:
      case 14:
         if(dont[i]==dont[j])
          if(t=onediff(m[i],m[j]))
          {
           checked[i] =
           checked[j] =
           changed    =
            newd[ncnt] = dont[j] | t;
            new[ncnt++] = m[j] & ~t;
          }
     }
    break;
  case 7:           /*  3-bit group   */
  case 11:
  case 13:
  case 14:
     for(j=0;j<terms;++j)
      if(m[j] == 15)     /* 4-bit group */
       if(dont[i]==dont[j])
        if(t=onediff(m[i],m[j]))
        {
         checked[i] =
         checked[j] =
         changed    =
          newd[ncnt] = dont[j] | t;
          new[ncnt++] = m[j] & ~t;
        }
  }
}
for(i=0;i<terms;++i)   /* keep unchecked terms */
 if(!checked[i])
 {
  saved[1][savcnt] = dont[i];
  saved[0][savcnt++] = m[i];
 }
/*  Remove duplicates */
for(i=0;i<ncnt-1;++i)
 for(j=i+1;j<ncnt;++j)
  if(new[i]==new[j] && newd[i]==newd[j])
   {
    new[j] = DELETE;
     changed = 1;
   }

 /* copy new[] back into m[] */
 for(terms=i=0;i<ncnt;++i)
 if(new[i]!=DELETE)
  {
   m[terms]=new[i];
```

```
        dont[terms++]=newd[i];
      }
    }
    while(changed);

    /* pick up saved terms */
    for(i=0;i<savcnt;++i)
    {
     m[terms] = saved[0][i];
     dont[terms++] = saved[1][i];
    }

}

/*****************************************
*
*  Special I/O interface with FLEX for
*  use with Heathkit H19 Terminal.
*  These routines interface directly with
*  FLEX's input/output to avoid echo and
*       system overhead.
*
*****************************************/
struct flexio {
       int (* inchne)();    /* Input no echo */
       int (* ihndlr)();    /* Int Handler */
       int (* swivec)();    /* Swi vector    */
       int (* irqvec)();    /* Irq vector    */
       int (* tmoff)();     /* Timer off     */
       int (* tmon)();      /* Timer on      */
       int (* tmint)();     /* Timer init    */
       int (* monitr)();    /* Monitor address */
       int (* tinit)();     /* Terminal init */
       int (* stat)();      /* Input status  */
       int (* outch)();     /* Output char   */
       int (* inch)();      /* Input w/echo  */
        } ;
struct flexio *iop = 0xd3e5;    /* I/O Table Pointer */
/*
***  getkey and outchar:
*
*    NOTE:  These routines are dependent on INTROL-C
*       Version 1.6 in that the D-register is used
*       as the first parameter of a function and
*       D-register is returned as the integer
*       result of a function.  Check other 6809 'C'
*       compilers or use '#asm' constructs for these
*       two routines.
*
*/
getkey() /* return parity stripped A-register */
{
   return(((* iop->inchne)()>>8)&0x7f);
}

outchar(c)
int c;
{
   (* iop->outch)(c<<8);  /*  Output A-register  */
}

/*
***  move(y,x);
*
*    y=line #   (0-23)
*    x=column # (0-79)
*/
move(y,x)
unsigned y,x;
{
 outs("\033Y");
 outchar(y+32);
 outchar(x+32);
}

/*
***  Erase screen
*/
erase()
{
 outs("\033E");
}

/*
*** printat --
*/
printat(y,x,s)
int y,x;
char *s;
{

   move(y,x);
   outs(s);
   move(y,x);

}

/*
**** output a string
*/
outs(s)
char *s;
{

   while(*s)
    outchar(*s++);

}


+++
```

BIT-BUCKET

By: All of us.....

The Editor,
Computer Publishing Center,
68 Micro Journal,
5900 Cassandra Smith,
P.O. Box 849,
Hixson, TN 37343.
U.S.A.

Chemistry Department
University of Transkei
Private Bag X5092
UMTATA
Republic of Transkei
Southern Africa

Dear Don,

Thank you very much for publishing my two programs, GENPL9 and FILECOMP, as well as my letter. I have enjoyed the single-board computer reviews very much. I hope any readers who build these systems will write in with their experiences, suggestions and add-ons. What has happened to the Artisan SBC?

I am submitting two programs for publication.

Program RENUMBER.BAS:

Although the TSC utility RENUMBER.CMD is pretty good for renumbering TSC Basic programs, it does have its limitations. Firstly, it fails to handle ERL= statements. It is frustrating to have to manually renumber ERL assignments and it is also very easy to make mistakes! Secondly, it does not allow partial renumbering of program segments, a useful feature to facilitate breaking up a program into logical blocks, or to move a block elsewhere.

To get around these problems I decided to write my own RENUMBERing routine. I wrote the program in TSC XBASIC to get the logic right, meaning to translate it into assembler later, but as it turned out to be fast enough for most needs and easy to modify, I left it in Basic. It is a disk-to-disk utility. That means that it reads a Basic file from disk and writes the output (optionally) to disk, whereas the TSC Renumber works on a Basic program in memory.

The program has several options: (a) renumber the whole program; (b) renumber only parts of the program; (c) renumber, using new parameters from a disk file instead of having to type them in. The last feature is especially useful with long programs, as it is much easier to use a word-processor to create the table of numbers, and the table can be easily edited if the output is not satisfactory. Any fatal overlap is optionally reported, as are missing target lines (e.g.: 'GOTO 800' withuut a line 800).

Output can be directed to the terminal, a printer or a disk file.

Program DATTR.CMD:

In one of your recent issues a contributor published a neat little program for setting or clearing the file protection codes of a Flex disk, which gave me the solution to one problem I had. I do a lot of correspondence, both for the Department and for myself, and I keep a disk just for these letters. The problem is, how to mark the file directory so that I can tell, from its listing, which letters have been replied to and which replies are pending. The solution was to CATalog protect all files that have been replied to, so that by running a DIR all the files with 'C' in the protection codes where clearly demarcated. I then modified TSC's DIR command so that CATalog protected files do not appear in the listing, and things were even easier. (Note CAT.CMD does this but does not display the date when the file was created, so is not as useful for this particular purpose).

I then set about writing a program that would set any combination of the protection codes, or clearing the lot, using the idea from that published listing. The program DATTR.CMD does just that. It has a few useful features: (a) it functions just like DIR.CMD, in that any files that match the given specifications in the command line are processed; (b) a wild card character can also be used (wild=?), as many times as required; (c) a HELP screen explains all the options if the command DATTR is used without any parameters; (d) an automatic repeat is available, otherwise a prompt is given for each file; (e) existing protection codes can be optionally retained, with the new ones added in; (f) at any time the process can be aborted, with a return to Flex, using 'Q' as a response to the prompt, or the ESCape key in auto mode.

I hope you will find these two programs up to standard and that other readers will find the programs per se, or at least the ideas behind them, useful. If anyone does not feel like typing out the listings of these or any other programs of mine that have been published, I can supply the source on 5" Flex9 disks if they send me $15 to cover the cost of the disk and postage. Please specify format (SSSD, DSDD, no. of tracks, etc.).

This is going to be a long letter. Since you like to receive the material on disk, what with the expense involved, I tend to accumulate things to say before I go ahead and write the whole lot. Please feel free to edit this letter anyway you like, or publish it in parts.

Since my last communication with you I received a letter from Microflex in Quebec in response to my letter in 68 Micro (Dec 1984, page 49) from someone who knows what happened to the DP09. Apparently the owner of Artisan Systems (Mike Kelly) has discontinued doing business with the DP09, which is a shame as the product looked really super from the advertisement and data sheets I was sent. It's all very true what you say, Don, that it's risky doing business with a company that may not be around for long, but how is one to know that. If everybody waited say a year before buying from XYZ, while company XYZ advertised in 68 MJ, by that time the company would be bust.

Recently I sent you a couple of programs one of which was a more elaborate version of the one submitted by Joseph Aulicino in June 1984, (thank you Joseph for the idea), which set/cleared the disk-file

protection bits in the Flex directory. In his article he mentioned a patch to Flex to make the FMS function #8 work. At that time I immediately made the correction to the FLEX.SYS on my current System disk and left it at that. It's interesting to note that I had never noticed that bug earlier, since I always seem to have used functions #9 and #10 to manipulate the directory. Anyway my program DATTR.CMD worked fine. Then in July this year one of my 40-track drives broke down. The only working spare drive I had was 80-track so I installed it as drive #1, leaving the other 40-track drive as #0. Now I had to use SYSGEN.CMD in order to have the proper drive control blocks set up for one 80-track drive, with a 3 ms step rate. To do this I got out my backup master disk and reconfigured the FLEX from the original supplied with my Gimix.

Everything worked fine until I went to use DATTR.CMD on my correspondence disk. The program just would not step to the next directory entry! Disaster, I thought, I have sent Don a program with a BUG! I got out the text file and looked through it but I just couldn't figure out where I had made the slip. I, of course, had forgotten that Joseph Aulicino's patch had not been implemented on the master FLEX copy! I was reminded when the program worked perfectly after booting the old 40/40 track system disk. So, readers, the patch works and is worth doing. It is amazing that an old and well established product like FLEX still has bugs in it!

Now I couldn't remember which bytes had been changed, and which issue of 68 Micro had the data. I looked through all the old journals (3 years' worth) several times, except of course the June 1984 copy which was sitting on top of the Gimix (courtesy of Murphy's Law)! Eventually I found it but this led me to write the program which I had been meaning to code for a long time, LCOMPARE.CMD, and this is the reason for all my ramblings above - to tell you that I am sending you this program which compares 2 files (any type), byte by byte, and displays any differences.

It makes sense to use the program only when small changes have been made, such as after using FIX.CMD, Sleuth or SAVE.CMD on an in-memory modified file. The display halts when a difference is found and displays both lines with the differences marked. Pressing any key resumes the display, but the 'A' key now disables the 'HALT'. Similarly, when 56 lines have been displayed, scrolling stops until the user presses a key. The 'A' response again disables the page pause. The 'LOCUS' is the byte number of the first byte in that line relative to the beginning of the file. If the two files get 'out of step' then all the subsequent lines will be flagged as different and the 'Q' key or ESCape key will get one out, back to Flex. I am working on a binary version which will display the actual load addresses.

By the way, I discovered that the Flex FMS function #5 (rewind file) does not honour the space compression flag set after the previous OPEN, but it must be explicitly recoded after using FMS function #5.

I am submitting a revised version of the TSC ECHO.CMD, called SPECHO. It does not use the routine NXTCHR for processing the line buffer so every character is faithfully sent to the output path. NXTCHR eats up spaces so that multiple spaces look like one space and I didn't like the way the old ECHO processed my Startup messages.

Now for another hopefully useful patch. My TSC Macro-assembler produces binary output to a file whose extension defaults to .BIN unless another extension is actually typed in the command line. Now I usually want to produce .CMD files, so I had a look at the code responsible for setting the default extension and I include a patch to reverse this.
To implement the change, at or around address $0205, find the first batch of code below. Note the address of the $4F (CLRA) instruction.
Make sure that YOUR 'offset,U' is $37,U otherwise change that number in my code to whatever your offset is in your version.
Now there are several ways to make the patch but make sure you have a backup copy of the original, just in case!

1. Use FIX.CMD -
   FIX <drive.>ASMB.CMD
   the : prompt will appear.
   Type M 0205 (or whatever address you found for $4F)
   That address plus the memory contents will be displayed. Just type in the new value against each location as shown.

```
Address old  new
   0205  4F  86
   0206  6D  02
   0207  C8  60
   0208  37  C8
   0209  26  37 (check if the same
                 in your version!)
   020A  02  26
   020B  86  01
   020C  02  4F
```

   Now press RETURN and then E - job is done.

2. Find the transfer addresses with MAP.CMD.
   Assemble a file such as the one below (* CHANGE TO *) with the proper origin, and a transfer address (mine is $0000), i.e. include the items in parentheses but without the parentheses.
   Then append this binary file to a COPY of your ASMB.CMD.

3. You can do an in-memory alteration and then use SAVE.CMD.
4. Use SLEUTH if you have it.

```
                * ASMPATCH
                * To change TSC Macro
                * Assembler's default
                * binary file output
                * from .BIN  to  .CMD



                * **** ORIGINAL HERE ****

0205                    ORG     $0205

0205            Z0205
0205 4F                 CLRA            deflt=.BIN
0206 6D C8 37           TST     $37,U   = $FF
                * (check if $xx,U is the same
                * in your version!)
0209 26 02              BNE     Z020D
020B 86 02              LDA     #$02    set .CMD
020D BD CD33    Z020D   JSR     >SETEXT

                ********************************
```

```
                * **** CHANGE TO ****

                      (ORG    0)
                (START  EQU    *)

0205                  ORG    $0205

0205            Z0205
0205 86 02            LDA    #$02   deflt=.CMD
0207 6D C8 37         TST    $37,U  = $FF
                * (check if $xx,U is the same
                * in your version!)
020A 26 01            BNE    Z0200
020C 4F               CLRA          set .BIN
020D BD CD33    Z020D  JSR    >SETEXT
                      (END START) transfer address
```

What I haven't been able to figure out is: under what conditions will $37,U be clear so that a .CMD extension is used by the assembler? Any ideas anyone?

Next, this is for anyone doing stock control in a laboratory environment. I have developed a package, running under TSC XBASIC precompiler, to keep control of chemical stocks, check for low stock, usage of each item, etc. I know there are a lot of readers who work in laboratories so if anyone is interested in this program they can write to me. It's free except for the cost of a disk and postage! I must know the disk formats though, and you must have the Extended Basic Precompiler.

Finally a few bugs and requests. Firstly, has anyone implemented a clock/calendar for the Sardis SBC such that the Flex DATE prompt can be eliminated? An article in 68 MJ would be nice on this topic.

Secondly, I have a second-hand CT-30 cassette interface but no documentation or schematics. If anyone can help I'll gladly refund postage and copying costs. Thirdly, (this is for those with the TSC Flex Diagnostics package), I have noticed that if I use VALIDATE.CMD soon after a fresh boot the program works perfectly, but after Flex has been used a lot without re-booting, the program aborts immediately with a string of messages such as:

```
ADDRESS: 0005, CRC ERROR
ADDRESS: 0005, CRC ERROR
FILE CONFLICT: DIRECTORY/DIRECTORY
FILE CONFLICT: FREE CHAIN/FREE CHAIN
DISK ERROR #9
```

After this the system is dead and requires a reboot. Anybody figure out the cause of all this?

Has anyone ever got printer-spooling working under Flex on a Gimix? I would be obliged for details as I have not been successful.

I have found a couple of bugs in my version of Stylograph. The first involves the 'find-and-replace' function. Say one wants to change the number 334 to 434 and uses the 'replace 3 with 4' method. After Stylo has successfully changed the first '3' and prompts for the second one, if the response is no, it just keeps doggedly on that very same character wanting to replace it. Try it. This may seem a trivial example but it becomes a nuisance when renumbering part a Basic program by hand.

The second bug involves printer codes. Assume that '/' is the printer code delimiter. Then sending the following text to my Prowriter turns out as shown:

(original)=The bug in Stylo\27,33\ here
(printout)=The bug in Stylo re

It seems to 'eat' the one or two characters following the delimiter! Putting a ghost hyphen between the delimiter and the following text solves the problem, but it is still a nuisance. Has anyone else seen these bugs?

Sorry Don that this has turned out to be such a long letter. I think there are a lot of new Flex users out there who may be happier with detailed procedures. I know that when I first started I really did need that kind of help. However if you think that the explanations for using FIX, etc., are unnecessary then take them out. That's what word processors are for anyway.

Yours faithfully,

Peter

(Dr L. Piacenza)

Editor's Note: Peter, don't fret about the long letter, we thrive on such as this. Your willingness to share and contend with the problems we have had with the postal services either here or there, has been a real deamon. Thankfully we now have it all under control and I and thousands of reader/users thank you for sharing your efforts and "sweat of the brow". Thanks Peter!!!!!

To my reader: Dr. Piacenza has also submitted some very nice utilities to S.E. Media for resale. These are now being "beta tested" and hopefully available soon. It is acknowledged that these "freebe" programs, presented here, could have also been included in the sale package. But he kindly offered them as articles - to share with all of us. I guess that is what we have really been all about these years past. Thanks to all of you, we are surviving' So to you Peter and all those others, from all of us, to all of you, THANKS also!!

DMW

- - -

```
1000    REM RENUMBER ON DISK - by LPL Piacenza.
1020    REM Will renumber a BASIC program on disk, to
disk/terminal/printer.
1040    REM capable of renumbering the whole program, or selected
blocks.
1060    REM
*************************************************************
1080    REM IMPORTANT - assumes each line number is followed by a
space!
1100    REM
*************************************************************
1120 NSZ=50:REM number of allowed sub-sections.
1140 DIM NX(2000),NZ(2000) : REM DIM>no. of program lines.
1160 DIM NFZ(100) : REM NFZ=line references not found.
1180 DIM STZ(NSZ),ETZ(NSZ):REM where to start, where to end per
block.
1200 DIM NSZ(NSZ),INCZ(NSZ):REM new start, new increment per block.
1220    REM ********************
1240 NS$="Maximum segments permitted ="
1260 INPUT"Filename to be renumbered (defaults to .BAS) ";FS
1280 ZZ$=FS:GOSUB 3920:FS=ZZ$
1300 PRINT"Output to T(erminal), P(rinter) or D(isk) ";
1320 GOSUB 4760:OU$=CA$
1340 OUZ=0:IF OU$="P" THEN OPEN "O.PRINT.SYS" AS 0
1360 IF OU$<>"D" THEN 1440
1380 OUL=2:INPUT"Output Filename ",FW$
1400 ZZ$=FW$:GOSUB 3920:FW$=ZZ$
1420 OPEN NEW FW$ AS 2
1440 OPEN OLU FS AS 1
1460    REM ********************
1480    REM ensure min. and max. for renumbering whole program.
1500 FOR IZ=1 TO NSZ:STZ(IZ)=0:ETZ(IZ)=32767:NEXT IZ
1520 LFZ=0
1540 SZ=1
1560    REM ********************
1580 PRINT"Renumber the WHOLE Program (Y=/N)? ";:GOSUB 4760:WH$=CA$
1600 IF WH$<>"N" THEN 1780
1620 PRINT"Enter data from a disk file (Y/N=)? ";:GOSUB 4760:DF$=CA$
1640 IF DF$="Y" THEN GOSUB 4800:GOTO 1900
```

```
1660     REM ********************
1680 PRINT"For each block to be renumbered, enter the requested
information"
1700 IF SZ>BSZ THEN PRINT:PRINT BS$;BSZ:GOTO 1900
1720 PRINT:INPUT"Enter STARTING and ENDING numbers (X,Y) ",STS,ETS
1740 STZ(SZ)=VAL(STS):ETZ(SZ)=VAL(ETS)
1760 IF ETZ(SZ)<STZ(SZ) THEN 1720
1780 INPUT"Enter New number, increment ",NNS,ICS
1800 NSZ(SZ)=VAL(NNS):ICZ(SZ)=VAL(ICS)
1820 IF WHS<>"N" THEN 1900
1840 PRINT"More sequence to renumber? (Y/N) ";:GOSUB 4740:BPS=CAS
1860 PRINT:IF RPS="Y" THEN SZ=SZ+1:GOTO 1700
1880     REM ********************
1900 PRINT
1920 ON ERROR GOTO 3800
1940 PRINT #0,SPC(15);"Renumber of program ";FS:PRINT #0
1960 FOR CYZ=1 TO SZ
1980 PRINT #0,"Starting from line ";STZ(CYZ);
2000 PRINT #0,TAB(40);"Ending at line  ";ETZ(CYZ)
2020 PRINT #0,SPC(2);
2040 PRINT #0,"New start - ";NSZ(CYZ);TAB(40);"New increment -
";ICZ(CYZ)
2060 NEXT CYZ
2080 PRINT #0
2100     REM ********************
2120 IF WHS<>"N" AND OFS<>"Y" THEN 2200
2140 PRINT"Check for overlapping blocks (Y/N)? "
2160 GOSUB 4760:OVS=CAS:PRINT
2180     REM ********************
2200 IZ=1
2220 PRINT "Setting up OLD NUMBER/NEW NUMBER table."
2240 INPUT LINE #1,A$
2260 NZ(IZ)=VAL(LEFT$(A$,INSTR(1,A$," ")))
2280 FOR JSZ=1 TO SZ
2300 NCZ=NSZ(JSZ)-ICZ(JSZ)
2320 IF NZ(IZ)<STZ(JSZ) THEN NEXT JSZ:GOTO 2520
2340 IF NZ(IZ)>ETZ(JSZ) THEN NEXT JSZ:GOTO 2520
2360     REM Fall through when line num is in selected block.
2380 NZ(IZ)=NCZ+ICZ(JSZ)
2400 NCZ=NZ(IZ)
2420 NSZ(JSZ)=NZ(IZ)+ICZ(JSZ)
2440 IZ=IZ+1
2460 GOTO 2240
2480     REM ********************
2500     REM Jump here when line num not in selected block.
2520 NZ(IZ)=NZ(IZ)
2540 IZ=IZ+1
2560 GOTO 2260
2580     REM ********************
2600 CLOSE 1
2620 TTZ=IZ-1:PRINT
2640     REM tables ready.
2660 IF OVS="Y" THEN GOSUB 4600
2680 IF OVS<>"X" THEN 2760
2700 PRINT"Abort or Carry on anyway (A/*C) ";:GOSUB
4760:OVS=CAS:PRINT
2720 IF OVS="A" THEN END
2740     REM ********************
2760 ON ERROR GOTO 3840
2780 OPEN OLD FS AS 1
2800 FOR LZ=1 TO TTZ
2820 AXS=""
2840 INPUT LINE #1,A$
2860     REM change the line number
2880 A$=MID$(STR$(NZ(IZ)),2,LEN(STR$(NZ(IZ)))-
2)+MID$(A$,INSTR(1,A$," "))
2900     REM vector verbs.
2920 YZ=1
2940 XS="ON":XZ=INSTR(YZ,A$,XS):IF XZ<>0 THEN QZ=0:GOSUB 4020
2960 XS="GOTO":GOSUB 3320
2980 XS="GOSUB":GOSUB 3320
3000 XS="THEN":GOSUB 3320
3020 XS="ELSE":GOSUB 3320
3040 XS="RESUME":GOSUB 3320
3060 XS="RESTORE":GOSUB 3320
3080 XZ=INSTR(1,A$,"ERL"):IF XZ=0 THEN 3140
3100 ZZ=INSTR(XZ,A$,":"):IF ZZ=0 THEN 3140
3120 XS=MID$(A$,XZ,ZZ-XZ+1): GSUB )))0
3140 PRINT #WUZ,A$;NXS
3160 NEXT IZ
3180     REM ********************
3200 CLOSE 1:IF OWZ<>0 THEN CLOSE OWZ
3220 GOSUB 4620
3240 END
3260     REM all done!
3280     REM *********************
3300     REM find vector verbs.
3320 YZ=1
)340 XZ=INSTR(YZ,A$,X$):IF XZ=0 THEN RETURN
)360 ZZ=XZ+LEN(X$):YZ=XZ+1
)380 CZ=XZ
)400 LLZ=VAL(MID$(A$,CZ)):IF LLZ=0 THEN ))40
)420 LLS=STR$(LLZ):LLS=MID$(LL$,2,LEN(LLS)-2)
)440 ZZ=CZ+LEN(LLS)
)460 GOSUB 4220
)480 @OSUB 3620
)500 IF ZTZ=0 THEN GOSUB 4)20:GOTO ))40
)520 NNS=" "+MID$(STR$(NZ(ZJZ)),2,LEN(STR$(NZ(ZJZ)))-2)
)540 A$=LEFT$(A$,XZ-1)+ES+NMS+MID$(A$,ZZ)
)560 GOTO )340
)580     REM ********************
)600     REM BINARY SEARCH OF NUMBER TABLE.
)620 ZUZ=TTZ+1:ZL3=0:ZJZ=ZUZ
)640 FOR ZZZ=0 TO 1 STEP 0
)660 ZCZ=ZJZ
)680 ZJZ=INT((ZUZ+ZLZ)/2)
)700 IF ZJZ=ZCZ THEN ZFZ=0:RETURN
)720 IF LLZ<NZ(ZJZ) THEN ZUZ=ZJZ:NEXT ZZZ
)740 IF LLZ=NZ(ZJZ) THEN ZFZ=1:RETURN
)760 ZLZ=ZJZ:NEXT ZZZ
)780     REM ********************
)800 IF ERR=8 THEN RESUME 2600
)820 ON ERROR GOTO
)840 IF ERR=8 THEN RESUME )200
)860 ON ERROR GOTO
)880     REM ********************
)900     REM append .BAS extension if necessary.
)920 XZ=INSTR(1,RIGHT$(ZZS,4),".")
)940 IF XZ<>0 THEN RETURN
)960 ZZS=ZZS+".BAS":RETURN
)980     REM ********************
4000     REM process CASE (ON verb).
4020 QZ=QZ+1:IF QZ>=LEN(A$) THEN RETURN
4040 IF MID$(A$,QZ,1)<>"," THEN 4020
4060 LLZ=VAL(MID$(A$,QZ+1)):IF LLZ=0 THEN 4020
4080 @OSUB 3620
4100 IF ZFZ=0 THEN GOSUB 4)20:GOTO 4020
4120 ZZ=QZ+1:GOSUB 4220:NMS="
"+MID$(STR$(NZ(ZJZ)),2,LEN(STR$(NZ(ZJZ)))-2)
4140 A$=LEFT$(A$,QZ)+NMS+MID$(A$,ZZ)
4160 GOTO 4020
4180     REM find each vector number, skipping leading spaces.
4200     REM ********************

4220 IF MID$(A$,ZZ,1)=" " THEN ZZ=ZZ+1:GOTO 4220
4240 IF MID$(A$,ZZ,1)>="0" AND MID$(A$,ZZ,1)<="9" THEN ZZ=ZZ+1:GOTO
4260
4260 RETURN
4280     REM ********************
4300     REM line references not found
4320 LFZ=LFZ+1
4340 NFZ(LFZ)=LLZ : AXS=" : REM????"
4360 RETURN
4380     REM ********************
4400     REM print all line references not found.
4420 IF LFZ=0 THEN RETURN ELSE PRINT
4440 PRINT"The lines followed by ': REM????' were not found."
4460 PRINT"Table of lines not found:":PRINT
4480 FOR IZ=1 TO LFZ
4500 PRINT NFZ(IZ)
4520 NEXT IZ
4540 RETURN
4560     REM **********************
4580     REM check for fatal overlap.
4600 FOR IZ=2 TO TTZ
4620 FOR IZ=2 TO TTZ
4640 IF NZ(IZ)<=NZ(IZ-1) THEN GOSUB 4700
4660 NEXT IZ
4680 RETURN
4700 PRINT  #0, "Fatal overlap at lines: ";NZ(IZ-1);"-";NZ(IZ)
4720 OVS="X" :RETURN
4740     REM ********************
4760 CAS=CHR$(ASC(INKEY$+0)) AND 95):PRINT:RETURN
4780     REM input from a disk file.
4800 PRINT:PRINT"The data file format must be:"
4820 PRINT"Starting line number, Ending line number"
4840 PRINT"New starting number, Increment"
4860 PRINT:PRINT"File FORMAT okay? ";:GOSUB 4760
4880 FMS=CAS:IF FMS<>"Y" THEN END
4900 WHS="Y"
4920 PRINT:INPUT"Name of file with values ",F$
4940 ON ERROR GOTO 5100
4960 OPEN OLD FVS AS 3
4980 INPUT #3,STZ(SZ),ETZ(SZ):REM Starting line number, Ending line
number
5000 INPUT #3,NSZ(SZ),ICZ(SZ):REM New starting number, Increment
5020 SZ=SZ+1:IF SZ>BSZ THEN PRINT BS$;BSZ:GOTO 5060
5040 GOTO 4980
5060 CLOSE 3:SZ=SZ-1:PRINT:RETURN
5080     REM ********************
5100 IF ERR=8 THEN RESUME 5060
5120 ON ERROR  OTO
```

To Be Continued Next Month

Dear Don,

I notice that my subscription is due to run out in another month, so I'm rushing this letter a little sooner than I intended, to renew for another 2 years. Wouldn't want to miss an issue!

Now for a little more in the continuing saga of XBASIC. First off, I'd like to expand a little on the short Decimal-to-HEX routine published in the May issue. Did anyone notice that the self-same program can be used to convert from decimal to **any** base in the range 2 thru 63? Though there'll be some pretty weird-looking digits beyond base 27!! All you have to do is to change the two occurrences of '16' in Line 30 to the desired base-number. Of course, if the **maximum** base-number never exceeds 10, you won't need the piece which adjusts by 7 for digits in excess of 9. Or, to really round off the program, change Line 10 to read

```
10  N$="": INPUT "Enter decimal-number and base",D,BX
```

and change '16' to 'BX' in Line 30.

It might be a good idea while I'm on the subject to give the reverse program, that is, to convert from any base to decimal, so here goes with a quickie.

```
10  D=0: INPUT "Enter number and base",N$,BX
20 REM CONVERT NUMBER-STRING IN ANY BASE TO A DECIMAL
30  JX=LEN(N$): FOR IX=1 TO JX
40    KX=ASC(MID$(N$,IX,1)-48: IF KX>9 THEN KX=KX-7
50    D=D+KX: IF IX<>JX THEN D=D*BX
60  NEXT IX
70  PRINT D: GOTO 10
```

Using logic functions instead of 'IF - THEN', Line 40 could be re-written so :

```
40  KX=ASC(MID$(N$,IX,1)-48+7*(KX>9)
```

Note that I use +7 this time because I want to **subtract** 7 if (KX>9) is true, ie it becomes -1. See if you can similarly compress Line 50 using the same technique, and then combine 40 and 50 into one line. Before I leave this subject altogether, it might be worth mentioning that to convert from Base 1 to Base 2 (and get the Decimal equivalent along the way) the two programs should be chained. That is, convert from Base 1 to decimal, and then straight on to convert decimal to Base 2.

You'll find that routines which use power-functions like a^b to achieve base-conversions are apt to get a little unreliable, as they use logs and antilogs in their calculations.

OK, enough of that. Let's move on to another area that seems to puzzle a few people, and that is when to use ASC, VAL, CHR$ or STR$. The definitions given in the XBASIC Manual really only have meaning for someone who already understands them, so a little explanation is probably in order here. We'll consider them as 2 sets of twins - one pair being ASC and CHR$, and the other VAL and STR$.

ASC(I$) will return the decimal ASCII value (not HEX) **of the first character in I$**, so if I$="Hello" then ASC(I$)=72 (the ASCII value of 'H'). But if coupled with the use of MID$, as in my example program above, or RIGHT$, it can be used to give the decimal equivalent of **any single character** in the string. Thus ASC(RIGHT$(I$,1))=111 which is the decimal ASCII value of "o". CHR$(I%) is the exact opposite, in that it generates the ASCII character equivalent of I%, so that CHR$(72), if outputted to the screen would display 'H', while CHR$(111) would display 'o'. But keep in mind that these two can handle only one character at a time.

VAL and STR$, on the other hand, can cope with multi-character strings. But, and a big but, they deal only with numeric strings. So, with I$="Hello" and J$="123.4", VAL(I$)=0 because 'Hello' is not a decimal number, but VAL(J$)=123.4. Similarly, given I=123.4, then STR$(1)="123.4". Big deal!, you say. So what purpose do they serve if they can only handle decimal numbers, and apparently leave these numbers unchanged?

Perhaps the best way to look at these is to imagine that VAL is a machine capable of transforming a picture of something into the real thing, while STR$ is an instant camera. So, if we had a short-tailed, black-spotted dog from which we wished to produce a long-tailed, zebra-striped pup, we would first of all take a picture of the original, then touch up the photograph appropriately, and finally use our VAL-machine to produce the desired animal.

In the same way, we would use STR$(I) to manipulate 'I' in ways that would be difficult, if not impossible, to do directly on 'I'. For example :

```
10 I=123.45: PRINT "You have $";I
```

RUNning this program would produce 'You have $ 123.45' with a space between the '$' sign and the '1', where an imaginary '+' sign resides. Here we have a problem

if we wish to produce '$123.45' from '$ 123.45'. Hold on though. Here comes $UPERSTR$ to the rescue, and we don't even need VAL. We re-write thus :

```
10 I=123.45: PRINT "You have $";MID$(STR$(I),2)
```

What we've done here is to convert 'I' to STR$(I) (that is, we've changed 123.45 into a string of ASCII characters corresponding to the individual digits of I), and then used MID$ to print out these characters, commencing at Character-2 in order to skip the imaginary '+'.

Or, to demonstrate an even trickier example, suppose we wished to reverse the order of the digits - no matter what the length of the number. We would write something like :

```
10  J$="": INPUT I: I$=STR$(I)
20  FOR J%=2 TO LEN(I$)
30  J$=MID$(I$,J%,1)+J$: NEXT J%: I=VAL(J$)
40  PRINT J$,I: GOTO 10
```

Line 20 begins looping at a count of 2 to skip the imaginary '+' sign, and I guess Line 40 would give a little cause for puzzlement, as it has apparently printed the same number twice. But don't forget that J$ is a picture-of-the-number, while 'I' is the actual number itself. Just as your reflection in a mirror (ignoring left-to-right reversal) may **look** like you, but it's not **really** you - only an image. You can do arithmetic with I, but not with I$.

VAL comes into its own in XBASIC programs which require numeric responses from the operator. How many times have you accidentally hit a letter (for example, the letter 'O' instead of the digit '0'), and had your program bomb with a 'Mixed Mode' error-message? Why not try something like :

```
10  INPUT "Enter a number ... ",I$: I=VAL(I$)
20  IF I=0 THEN PRINT "Invalid entry!": GOTO 10
```

This way, your program won't bomb, as an erroneous non-numeric entry will now produce a VAL of '0'. But what if '0' also happens to be a desired response? We won't want to keep bouncing back to Line 10 in that case! The solution is simply to insert a Line 15 :

```
15  IF I$="0" GOTO 30
```

There's only one slight complication here, and that occurs should you enter say 12B45 instead of 12345 (so maybe you've had just one teensy drink too many!). VAL(12B45) comes out as 12, as the conversion stops if a character is not a digit, a '+' or a '-' or a decimal-point. I'll leave you to ponder how to distinguish between such an erroneous '12' and a genuine entry of '12' (or any similar combination) until my next letter.

And that about wraps it up for this time. Keep those letters a-coming, though I can't guarantee a reply to each and every one.

Don Williams,
68 Micro Journal,
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,

Bob

R. Jones
President

1208 NW Grant
Corvallis OR 97330
May 29, 1986

68' Micro Journal
5900 Cassandra Smith Rd.
Hixson TN 37343

Dear Don,

I'm submitting some 6809 printer driver listings which might be useful. The two short ones are a "matched set" of resident ACIA and PIA drivers. Each fits in the Flex resident printer driver area, but still provides all four standard entry points. Both take their port address from PTDEV ($CC39).

The longer listing implements an irq-driven printer (ACIA) with a circular queue. If you have an old, slow printer and a program that uses a fair amount of cpu time deciding what to print, this can speed up output quit a bit. It needs to be appended to the normal P.COR for use, and also requires the irq routines provided in SWTP versions of Flex. It can easily be adapted for other irq conventions; anyone who needs help should feel free to write.

Yours truly,

Wilson M. Federici

```
*
* Flex ACIA printer driver,
* resident in Flex printer region,
* takes device address from PTDEV,
* provides all four entry points.
* wmf 4/20/86
*
* Wilson M. Federici
* 1208 NW Grant
* Corvallis OR 97330
*
CC39                    PTDEV   EQU     $CC39
*
*
* device address, alterable
CC39                            ORG     PTDEV
CC39 F47D                       FDB     $F47D
*
* $CCC0-CCCF initialize
CCC0                            ORG     $CCC0
CCC0   86      03       INIT    LDA     #3
CCC2   A7 9F CC39               STA     [PTDEV]
CCC6   86      15               LDA     #$15
CCC8   A7 9F CC39               STA     [PTDEV]
CCCC   39                       RTS
*
* $CCD0-CCD7 close
CCD0                            ORG     $CCD0
CCD0   86      0D       CLOSE   LDA     #$0D
CCD2   20      10 CCE4          BRA     OUTPUT
*
* $CCD8-CCE3 status
CCD8                            ORG     $CCD8
CCD8   34      02       STATUS  PSHS    A
CCDA   A6 9F CC39               LDA     [PTDEV]
CCDE   46                       RORA
CCDF   46                       RORA
CCE0   46                       RORA
CCE1   35      82               PULS    A,PC
*
* $CCE4-CCF7 output
CCE4                            ORG     $CCE4
CCE4   8D      F2 CCD8 OUTPUT  BSR     STATUS
CCE6   2A      FC CCE4         BPL     OUTPUT
CCE8   34      10              PSHS    X
CCEA   BE    CC39              LDX     PTDEV
CCED   A7 01                    STA     1,X
CCEF   35      90               PULS    X,PC
*
0000                            END
```

```
*
* Flex PIA printer driver,
* resident in Flex printer region,
* takes device address from PTDEV,
* provides all four entry points.
* wmf 4/20/86
*
* Wilson M. Federici
* 1208 NW Grant
* Corvallis OR 97330
*
*
CC39                    PTDEV   EQU     $CC39
*
* device address, alterable
CC39                            ORG     PTDEV
CC39 F400                       FDB     $F400
*
* $CCC0-CCCF initialize
CCC0                            ORG     $CCC0
CCC0   8D      1E CCE0  INIT    BSR     GETPRT
CCC2   CC    3AFF               LDD     #$3AFF
CCC5   A7 01                    STA     1,X
CCC7   E7 84                    STB     0,X
CCC9   C6      3E               LDB     #$3E
CCCB   4F                       CLRA
CCCC   E7 01                    STB     1,X
CCCE   20      18 CCE8          BRA     OUT1
*
* $CCD0-CCD7 close
CCD0                            ORG     $CCD0
CCD0   86      0D       CLOSE   LDA     #$0D
CCD2   20      1D CCE4          BRA     OUTPUT
*
CCD4   35      92       OUT2    PULS    A,X,PC
CCD6   12                       NOP
CCD7   12                       NOP
*
* $CCD8-CCE3 status
CCD8                            ORG     $CCD8
CCD8   34      10       STATUS  PSHS    X
CCDA   8D      04 CCE0          BSR     GETPRT
CCDC   6D 01                    TST     1,X
CCDE   35      90               PULS    X,PC
*
CCE0   BE    CC39       GETPRT  LDX     PTDEV
CCE3   39                       RTS
*
* $CCE4-CCF7 output
CCE4                            ORG     $CCE4
CCE4   8D      F2 CCD8  OUTPUT  BSR     STATUS
CCE6   2A      FC CCE4          BPL     OUTPUT
CCE8   34      12       OUT1    PSHS    A,X
CCEA   8D      F4 CCE0          BSR     GETPRT
CCEC   6D 84                    TST     0,X
CCEE   A7 84                    STA     0,X
CCF0   86      36               LDA     #$36
CCF2   A7 01                    STA     1,X
CCF4   86      3E               LDA     #$3E
CCF6   A7 01                    STA     1,X
CCF8   20      DA CCD4          BRA     OUT2
*
0000                            END

*
* IP.SYS
* irq-driven serial printer with buffer
* for use with P.COR
* wmf 2/86
*
* Wilson M. Federici
* 1208 NW Grant
* Corvallis OR 97330
*
*
* SWTP/Flex9 irq service routines
03E1                    SETIRQ  EQU     $03E1
03E3                    CLRIRQ  EQU     $03E3
*
C300                            ORG     $C300
C300 03FE                       FDB     DRVSIZ
*
C302   16    0018 C320  JOPEN   LBRA    IPOPN
C305   16    0041 C349          LBRA    IPCLS
C308   16    005E C369          LBRA    IPOUT
C30B   16    0043 C351          LBRA    IPCHK
* default port address, adjust to suit
C30E F470               ACIA    FDB     $F470
C310 00                 SIDE    FCB     0
C311 00                         FCB     0
*
* SWTP/Flex9 irq "information block":
*  - device address
*  - irq service address
*  - link space (2 bytes)
*  - status mask (1 byte)
C312                    IPBLK   RMB     2
C314                            RMB     2
C316 0000                       FDB     0
C318 8D                         FCB     $80
*
C319 00                 IPBSY   FCB     0
C31A                    QPUT    RMB     2
C31C                    QGET    RMB     2
C31E                    QFOLD   RMB     2
*
C320   30 8D 0080 C3A4 IPOPN   LEAX    IPSRV,PCR
C324   AF 8C   ED C314          STX     IPBLK+2,PCR
C327   30 8D 00A8 C3D3          LEAX    QSTRT,PCR
C32B   AF 8C   EC C31A          STX     QPUT,PCR
C32E   AF 8C   EB C31C          STX     QGET,PCR
C331   30 8D 03CB C700          LEAX    QLIM,PCR
C335   AF 8C   E6 C31E          STX     QFOLD,PCR
C338   E6 8C   05 C310          LDB     SIDE,PCR
C33B   58                       ASLB
C33C   58                       ASLB
C33D   AE 8C   CE C30E          LDX     ACIA,PCR
C340   3A                      ABX
C341   AF 8C   CE C312          STX     IPBLK,PCR
C344   86      03               LDA     #3
C346   A7 84                    STA     ,X
C348   39                      RTS
*
C349   1C      EF       IPCLS   CLI
C34B   6D 8C   CB C319  CLS1    TST     IPBSY,PCR
C34E   26      FB C34B          BNE     CLS1
C350   39                      RTS
*
*
C351   34      14       IPCHK   PSHS    B,X
C353   AE 8C   C4 C31A          LDX     QPUT,PCR
C356   30 01                    LEAX    1,X
C358   AC 8C   C3 C31E          CMPX    QFOLD,PCR
C35B   26      03 C36D          BNE     CHK1
C350   30 8C   73 C3D3          LEAX    <QSTRT,PCR
C360   AC 8C   89 C31C  CHK1    CMPX    QGET,PCR
C363   27      02 C367          BEQ     CHK2
C365   C6      FF               LDB     #$FF
C367   35      94       CHK2    PULS    B,X,PC
*
C369   34      14       IPOUT   PSHS    B,X
C36B   1A      10       OUT0    SEI
C36D   E6 8C   A9 C319          LDB     IPBSY,PCR
C370   26      16 C388          BNE     OUT2
C372   30 8C   9D C312          LEAX    IPBLK,PCR
C375   AD 9F 03E1               JSR     [SETIRQ]
C379   AE 84                    LDX     ,X
```

```
C37B  C6      35            LOB    #$35
C37D  E7 84                 STB    ,X
C37F  A7 01                 STA    1,X
C381  6C 8C   95 C319       INC    IPBSY,PCR
C384  1C      EF      OUT1  CLI
C386  35      94            PULS   B,X,PC
*
C388  AE 8C   8F C31A OUT2  LDX    QPUT,PCR
C38B  A7 80                 STA    ,X+
C38D  AC 8C   8E C31E       CMPX   QFOLD,PCR
C390  26      03 C395       BNE    OUT3
C392  30 8C   3E C303       LEAX   <QSTRT,PCR
C395  AC 8C   84 C31C OUT3  CMPX   QGET,PCR
C398  26      04 C39E       BNE    OUT4
C39A  1C      EF            CLI
C39C  20      CO C36B       BRA    OUTO
*
C39E  AF 8D FF78 C31A OUT4  STX    QPUT,PCR
C3A2  20      E0 C384       BRA    OUT1
*
C3A4  6F 8D FF71 C319 IPSRV CLR    IPBSY,PCR
C3A8  AE 8D FF70 C31C       LDX    QGET,PCR
C3AC  AC 8D FF6A C31A       CMPX   QPUT,PCR
C3B0  26      0B C3BD       BNE    SRV1
C3B2  86      15            LDA    #$15
C3B4  A7 A4                 STA    ,Y
C3B6  30 C4                 LEAX   ,U
C3B8  AD 9F D3E3            JSR    [CLRIRQ]
C3BC  3B                    RTI
*
C3BD  A6 80           SRV1  LDA    ,X+
C3BF  A7 21                 STA    1,Y
C3C1  AC 8D FF59 C31E       CMPX   QFOLD,PCR
C3C5  26      03 C3CA       BNE    SRV2
C3C7  30 8C   09 C303       LEAX   <QSTRT,PCR
C3CA  AF 8D FF4E C31C SRV2  STX    QGET,PCR
C3CE  6C 8D FF47 C319       INC    IPBSY,PCR
C3D2  3B                    RTI
*
C303                  QSTRT EQU    *
C700                  QLIM  EQU    $C700
* this yields maximum queue size, specifically:
0320                  QSIZE EQU    QLIM-QSTRT
* If smaller queue desired, replace
* queue equates with:
*    QSTRT RMB <size>
*    QLIM  EQU *
*
03FE                  DRVSIZ EQU   QLIM-JOPEN
*
0000                  END
```

OMNI PUBLICATIONS INTERNATIONAL LTD
1965 BROADWAY
NEW YORK, NY 10023-5965
TELEPHONE (212) 496-6100

FOR IMMEDIATE RELEASE **CONTACT: MARCIA POTASH**
BOB BAXTER

ATTENTION: DISK KILLERS ON THE LOOSE

In the June issue of OMNI writer Steve Gross warns home computer buffs to be on the look out for disk killing programs. Although little is known about disk killers outside of the computer industry, according to Gross a growing number of home computer users are learning about them the hard way.

Computer disk killers are programs designed specifically to destroy a computer's floppy and hard-disk drives. The killer programs, often disguised as free software, are placed anonymously on one or several of hundreds of computer bulletin boards across the country. Unsuspecting home users, hoping for a bargain, download the "free" program only to find out too late, often in front of a blank computer screen, what the killer program actually did. Moreover, there is usually no way to find out who placed the killer software, or from what location.

According to Gross, two of the nation's biggest computer networks now have programs available to detect killer software, as well as a list of current killer programs. Digital Dispatch, Inc., of Minneapolis calls its system Data Physician, and Compuserve offers C4Bomb. Both systems are designed to spot software disk killers before they have a chance to destroy your programs.

"He is smart, and knows it," says Gross about the programer who would design disk killers. "But he needs a forum to prove it to everyone else. The hundreds of computer bulletin boards in this country are his forum."

• • • •

SCULPTOR

SCULPTOR Release 1.14 — SUMMARY

General Improvements

- The Sculptor programs themselves require less disk space and less memory at run-time, particularly on MS-DOS
- The format of compiled sage and sagerep language programs is more compact, allowing larger programs to be run and improving program load time.
- Utilities are provided to convert old programs to the new format.
- There is more space in text which may be translated with the configuration program kcf.
- On MS-DOS, the following programs now accept wildcards in their filename arguments:
  describe, kicheck, killet, newkif.
- New vdu sequences have been introduced, including ten for user defined use.
- There are ten new, general purpose, printer control sequences.

Enhancements to sage and sagerep

- The exec command can now call a simple program directly without invoking a new shell.
- The chain command on MS-DOS now removes the old program from memory before running the new program.
- alphanumeric formats which previously applied only on input, may optionally be applied to assignments.
- Expanded alphanumeric formats allow formatted conversion of numeric and date fields into alphanumeric fields.
- New commands in sage and sagerep

getstr(source,pos,len)	Extract a sub-string
setstr(dest,pos,len,source)	Overwrite a sub-string
interrupts (on/off)	Enable/disable keyboard interrupts
clearbuf <file id>	Clear a file's record buffer
x ...	Extended alphanumeric statement
prev <file id> ...	Read previous record
pause	Wait for alarm interrupt
wakeup <task id>	Send alarm interrupt

The pause and wakeup commands are limited to Unix and similar operating systems.

Enhancements to sage

- A new special temp vdu returns the name field from the vdu parameter file.
- A control key may be defined in the vdu file to redisplay the background form.
- New commands in sage:

highlight <field list>	Display data with special highlight
at <y>,<x>	Position cursor to y,x
at <y>,<x> <string>	Display text at y,x
vdu <n>	Send vdu sequence number <n>

Enhancements to sagerep

- A new special temp printer returns the name field from the printer parameter file.
- The spc and tab functions now permit either a field or a constant argument.
- The zip trap is now available on file access commands.
- The following commands now available in sagerep as well as in sage:
  rewind, readkey, random, setdev.

Enhancements to menu

- Ten call line arguments d0 through $9 are available for substitution in command lines.
- The % and $ special characters may be escaped with a \.
- A menu file may include blank lines for readability.
- There is an option to inhibit a new shell as a simple command.
- interrupts are now reset to their default state before a program is called.

Modifications to Utilities

- kbrepq and kbformat now copy the last record even if its key is null.
- reformat does not display a count every 100 records unless the -c option is specified.
- vdumform no longer prints superfluous trailing spaces.

MDD

Micro Dynamics Limited, 1 Enfield Place, London NW9 5EE Tel 01 200 2277

SOMETHING FOR ALL OF US / FROM ALL OF US

CERTIFIED SOFTWARE CORPORATION

616 CAMINO CABALLO, NIPOMO, CA 93444 USA TELEPHONE 805-929-1395 TELEX 467013

Dealer/OEM Newsletter number 17

INTERNATIONAL DEALER/OEM TERMS

If you are an international dealer then you will find enclosed a new set of payment terms. This is similar to the one that went into effect last September, with the major change being the lowering of discounts that were intended to compensate for exchange rates. The dollar has come down considerably, and in some cases is considered to be lower than it should be. Should the dollar increase again to unreasonable levels, a new set of payment terms will be issued.

68000 COMPILER STATUS

The target debugger is undergoing testing by those companies who assisted me last month in Europe with porting to their systems. Those companies are : ELTEC, PEP, ELSOFT, and GESPAC.

Work is underway on drivers for the 68881 chip. After this is done then double precision reals will be added to the compiler and runtime. A few other improvements will be made in the software and then releases will be made of version 1.1 of the PCSK package. People with duplication licenses should return their master disks to receive the new versions as soon after release as possible. As a guide for returning your disks, you should expect version 1.1 to be available by :

OS-9/68000 August 1, CP/M-68K August 20, VERSAdos October 1

68000 COMPILER MANUALS

Since at least 80% of the 68000 compiler packages are sent to Europe, I have decided to start printing those manuals in DIN A4 format. These will probably be available around the middle of August, depending on how long the current supply of manuals lasts. The updated pages and OS dependent sections for version 1.1 will initially be printed in 8 1/2 X 11 format to work with the old manuals. Later, the manual will be reprinted using the newly acquired Hewlett-Packard LaserJet printer in A4 format. When this is done, you will be sent new OS dependent masters to be used when you start receiving the A4 manuals. The A4 manuals will be drilled with 4 holes for standard European binders.

There will be no change to the 6809 manuals, since the sales of these products are in a declining sales phase.

68000 DATA SHEETS

New 68000 data sheets will also be available by the middle of August. These will be multi-page, typeset, with diagrams, offset printed on glossy stock, a large improvement over the current data sheets.

OMEGASOFT™ PASCAL

GESPAC

GESPAC INTRODUCES A LOW COST, MEDIUM RESOLUTION GRAPHIC CONTROLLER FOR INDUSTRIAL APPLICATIONS.

MESA, AZ, June 15, 1986--GESPAC, Inc. introduces an inexpensive and powerful color graphics controller board. The GESVIG-3 is built on a single height Eurocard and is compatible with the standard G-64 bus. The G-64 bus is an easy-to-interface, non-multiplexed 16-bit bus specifically aimed at low to midrange industrial applications.

The GESVIG-3 is software programmable to operate in either European 50 Hz or American 60 Hz operations. The board can display at resolutions up 256 lines of 640 pixel. Each pixel is four bits deep thus allowing up to 16 different colors to be displayed at once in the same screen. An onboard color lookup chip allows each of these colors to be user defined as any one of a choice of 4096.

The GESVIG-3 uses a graphic processor capable of recognizing and executing several basic commands such as automatic vector drawing. The graphic processor and its on-chip character generator ROM is also capable of generating text in various sizes, orientations and styles.

The GESVIG-3 is directly connected in the I/O map of the G-64 bus, and can be used in conjunction with any of the 8 and 16-bit microprocessors available on the bus. The board can be connected to any monitor through its Red, Green, Blue and synchronization outputs.

GESPAC supports this board with GPS, an advanced graphics software package. GPS allows the user to easily translate program parameters into pictures using a complete set of high level commands.

The superior graphics capabilities of the GESVIG-3, and its stiff and compact board format, makes it ideal for such applications as public addresse displays, instrument consoles, industrial data terminals, and navigation computer displays.

The GESVIG-3 is available today at the low unit price of $990. The GPS graphic software package is available from GESPAC at $195.

For more information contact:

Joe Murphy
GESPAC, Inc.
100 W. Hoover Ave.
Mesa, AZ 85202
(602) 962-5559

Classifieds

Winchester 10 Megabyte Drives
Two (2) 10 Megabyte Hard-Disk Winchester Drives. Working - were removed for upgrade to larger drives.

1 - RMS Model #509 $275.00
1 - Seagate Model #412 $275.00

(615) 842-4600 Tom 9-5 EST.

LSI 68008 CPU card, "C" Compiler and Digital Research CPM/68K $350.

Tano Outpost II, 56K, 2 5"DSDD Drives, FLEX, MUMPS $595.

MICROKEY Single Board Computer, Target 128K RAM, FLEX, FORTH, with optional 6502 CPU & ROMS as advertised on p. 51 DEC. 84 68' Micro Journal. $1800.

1-PT-69 complete with Dual 5" DSDD Disk System and Controller, includes FLEX DOS. $745.

TELETYPE Model 43 PRINTER - with serial (RS232) interface and full ASCII keyboard. $359.00 ready to run.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09 CPU Card $1290.

1-CDS1 20 Meg Hard Disk System with Controller $1000.

(615) 842-4600 M-F 9 AM to 5 PM EST

•••

68008 HARD DISK SYSTEM - COMPLETE

512K 68008 system, 10 megabyte hard disk, Xebec 1410A HD controller, 80 track double side, double density floppy. Complete with cabinet/power supply. Taken in on Mustang-020 trade-in. Version 1.2 OS-9, Basic09, Stylo, Mail Merge, Spelling Checker, Dynacalc - like new - original price $2,900.00 range (advertised) - SPECIAL - ONLY $1750.00.
615 842-4600 - Data Comp, ask for Don or Tom.

•••

FOR SALE: GIMIX 6809 Level II System, 192kb Ram; 19mb HD; 800kb floppy; w/OS-9 Level II DOS, Pascal, Basic09, etc. Two ADDS Viewpoint 3-A Terminals; Anadex 1501 Printer. Best offer. Sam Ciammitti – (602) 996-9335 – Phoenix, AZ.

•••

Sardis ST-2900 system. CPU, FDC, two 5 1/4" Teac drives, power supply and cabinet. $500. (316) 685-7346.

•••

ATTENTION SWTP USERS - S/09 Upgradeable to S+, 64K Memory, 6 Serial Ports, 1 Parallel Port, Software, Manuals, Qume Terminal. Make Offer! Call (314) 469-3100.

•••

OS9/FLEX GIMIX 6809 Plus CPU Board; MMU Beyond 64K; Epson Printer. Two lowprofile TEAC Drives, 40T/80T DSDD; Heathkit H19 Terminal; Nice software. $1200 negotiable. Mike Lemon, 23 E. Belvue Road, Taylors, SC 29687.

•••

PT-69 system with three 80 track 5 1/4" DSDD drives and one 40 track 5 1/4" DSDD drive in two modern cases. With FLEX and application software. $550.00 Tom (206) 226-3708 evenings.

OS-9 UniFLEX
MUSTANG-020, 68020, 68881 AND MORE
HANDS-ON EXPERIENCE

The DATA-Comp Division of Computer Publishing Corporation announces their new and innovative HANDS-ON 68020 computer familiarization two day event. A chance to TRY BEFORE YOU BUY!

For two full days (Monday through Friday - excluding legal holidays) each participant will be furnished the exclusive use of a 68020 computer (MUSTANG-020). Each system will have available native C compilers, BASIC, assembler and other high level languages. Each system will be equipped with the Motorola MC 68881 math co-processor, where applicable.

Each demonstration room will contain not more than two work stations. Each system will be equipped with floppy disk, 20 megabyte winchester technology hard disk, and 2 megabyte of RAM. RAM is partitioned as 690K bytes of RAM disk and 1.2 megabyte of user RAM space.

Participants are encouraged to bring along any source level projects, for evaluation, in C, BASIC or assembler. Call for availibility of other HHLs.

Although this is not a training seminar, Data-Comp personnel are available for assistance and consultation. This event is scheduled for hands-on evaluations of the 68020 CPU, 68881 math co-processor and MUSTANG-020 system, operating in a functional environment.

Transportation to and from the airport and hotel/motel will be provided. Lunch provided both days. Chattanooga airport is serviced by American, Delta, Republic and other airlines.

COST

One person - $375.00

Two persons - $595.00

* Motel single $22.00, double $26.00
Includes satellite TV - convenient to food and shopping

DATA-COMP

A Division of
Computer Publishing, Inc.

5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

Systems available for both OS-9 and UniFLEX. Reservation should be made 15 days in advance. Attendee should initially indicate OS-9, UniFLEX or both. Special facilities available on request. Please write or call for additional information.

NOTE: Both OS-9 and UniFLEX are Unix type operating systems. Each as been enhanced in some aspect or another. Prospective attendees should have some working knowledge or experience with one of these operating systems, to gain full benefit of the session. However, a newcomer will find that it is a simple matter to be fairly proficient in using these systems in the allocated time. Special system instruction available on request. Call or write.

* Hotel/Motel cost are separate cost, not included in the basic cost shown.

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles
OS9 USER NOTES

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocesaor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASC

NOTE: .C1..C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
* All Currency in U.S. Dollars

Continually Updated In 68 Micro Journal Monthly

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615) 842-4601
Telex 5106006630

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

Coming Soon!

68000 products running under FLEX™

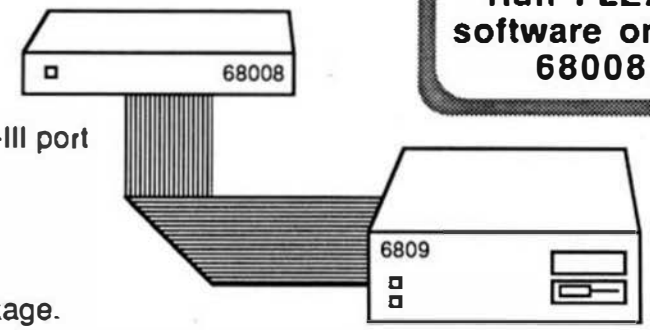
PLμS-68k (PL/9 for the 68000)

- Built-in screen editor
- Built-in source-level debugger
- Built-in assembler
- Byte, Integer, Long and Real variables
- Signed or unsigned variables
- Single-pass compiler
- Direct source to object
- Compiles over 1000 lines/min
- Requires second processor and any FLEX™ system with a PIA port

99% Compatible with PL/9

The second processor module (included with the compiler):

- 10MHz 68008 CPU
- 512K bytes RAM
- Case and power supply
- Plugs into Windrush UPROM-III port



Run FLEX™ software on the 68008

Interface software included:

- Program loader
- 68000 FLEX™ interface package.

Other Software:
68000 Assembler 68000 System Monitor
Programmer's Editor

For further information, phone or write:

Worstead Laboratories
North Walsham
Norfolk NR28 9SA
England

Tel (44) 692 404086
Telex 975548 WMICRO G

WINDRUSH
Micro Systems Ltd.

BOARDS & PARTS FOR GIMIX SYSTEMS

CONTACT GIMIX FOR PRICES, DETAILS, AND REQUIRMENTS FOR HARD DISK AND OTHER MASS STORAGE UPGRADES.

THE GIMIX CLASSY CHASSIS #19 consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Motherboard with Baud rate generator board $1498.19
#22 Triple Disk Regulator Card $88.22
#93 Baud Rate Generator Board $88.93
#23 Missing Cycle Detector $36.23
#92 Filler Plate ..... $14.92 50 Hz Option $30.00
Cable sets 8" with Back Panel connector $29.25
for two 8" external drives ..... $44.26 for two 5" drives $34.96

CPU BOARDS

#01 GMX III CPU & OS-9 GMX III $1898.01
#02 GMX IIICPU & UniFLEX III $1998.02
The #05 GIMIX 6809 PLUS CPU Board $578.65
Options: GIMIX DAT ..... $35.00 9511A $312.00
SWTP Cal ..... $15.00 9512 ..... $265.00
#03 6800 CPU $224.03
#06 6800 CPU w/Timers $268.04
6800 Baud Rate Option ..... Add $30.00

FLOPPY DISK CONTROLLER

#68 DMA $588.88

MEMORY BOARDS FOR 6809/68020 SYSTEMS

#72 256KB CMOS STATIC RAM board with battery back up (specify system) $848.72

MEMORY BOARDS (6800/6809 SYSTEMS ONLY)

#34 8K PROM Card $98.34
#32 16 Socket PROM/ROM/RAM Board, 24 pin $236.32
#31 16 Socket Universal Memory Board, 24/28 pin $268.31

INTELLIGENT I/O PROCESSOR BOARDS

significantly reduce systems overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K Baud. For use with GMX III and 020 systems.

#11 3 Port Serial-30 Pin (OS9) $498.11
#14 3 Port Serial-30 Pin (UniFLEX) $498.14
#12 Parallel-50 Pin (UniFLEX-020) $538.12
#13 4 Port Serial-50 Pin (OS9 & UniFLEX-020) $818.13
#15 24K Version of #11, with either large input or output buffers (specify) $648.15

I/O BOARDS (6800/6809 SYSTEMS ONLY)

#41 Serial, 1 Port $88.41
#43 Serial, 2 Port $128.43
#48 Serial, II Port (OS9/FLEX only) $318.48
#42 Parallel, 2 Port $88.42
#44 Parallel, 2 Port (Centronics pinout) $128.44
#45 Parallel, II Port (OS9/FLEX only) $198.45
#50 I/O for RS-232C, 423, 422-w/6850 ..... $244.50
#52 SS0A with 6852 ..... $258.52
#54 ADLC with 6854 ..... $268.54

CABLES FOR I/O BOARDS — SPECIFY BOARD

#95 Cable sets (1 needed per port) $24.95
#51 Cent. B.P. Cable for #12 & #44 ..... $34.51
#53 Cent. Cable Set $36.53

OTHER BOARDS & PARTS

#66 Prototyping Board-50 Pin $56.68
#33 Prototyping Board-30 Pin $38.33
Windrush EPROM Programmer S30 (OS9/FLEX 6809 only) $545.00
#76 Video Board-80 x 24 ..... $398.76
#08 Relay Driver Package $1128.08
#86 Above without Relays $538.86
Opto Board $348.85
Binder, 3" $12.00
Binder, 2" $9.00

8" DRIVE CABINET & PARTS

2 8" DSDD Drives, Cabinet & Cables 60 Hz only $1698.88
Cabinet Only for 8" Drive $848.18
220v/50 Hz. Option Add $30.00
Cable Set-Internal for 2 Drives $44.82
Cable Set-Internal for 4 Drives $67.64
Cable from 8" Cab. to Mainframe $45.81
8" Filler Plate $14.83

SOFTWARE:

GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the #08 controller. When ordered with controller, FLEX is $30.00
GIMIX versions of FLEX $90.00
GMX VDisk for FLEX 09 ..... $100.00
GMXBUG: PROMs & Manual $148.65
Boot or Video/Boot PROMS (6809) $30.00
GIMIX Boot PROM for UNIFLEX $50.00
RMS (OS9) $250.00
DD (OS9) $78.00
OS-9 GMX III Update w/CPU SPPTROM $125.00
I/O PROMS w/Update $40.00
GMXBUG/FLEX/VDISK w/OS-9III update ..... Add $175.00
RAM Disk for OS-9 ..... $125.00
O-FLEX $250.00
OS9 GMX I $250.00
OS9 GMX II $500.00
SCULPTOR-6809 (UniFLEX/OS-9) $995.00
SCULPTOR—68020 (UniFLEX) $1595.00

CONTACT GIMIX FOR PRICES AND AVAILABILITY OF OPTIONAL UNIFLEX AND OS9 LANGUAGES AND OTHER SOFTWARE.

ALL PRICES ARE F.O.B. CHICAGO.

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

LIMITED WARRANTY

GMX INC. ("GMX") Warrants its products against defects in material and workmanship for a period of ninety days from the date of shipment. The obligation of GMX is limited to the repair or replacement of any product, free of all charges, which proves defective during this period. This warranty does not cover damage due to accidents, negligence, abuse or tampering.

GMX MAKES NO OTHER WARRANTIES OR GUARANTEES, EXPRESS, STATUTORY, OR IMPLIED, OF ANY KIND WHATSOEVER WITH RESPECT TO ANY PRODUCT PURCHASED, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS HEREBY DISCLAIMED BY GMX AND EXCLUDED FROM ANY AGREEMENT MADE BY GMX.

GMX will not be responsible for any damage of any kind not covered by the exclusive remedies set forth in this limited warranty. GMX will not be responsible for any special, indirect, or consequential damage caused by its products.

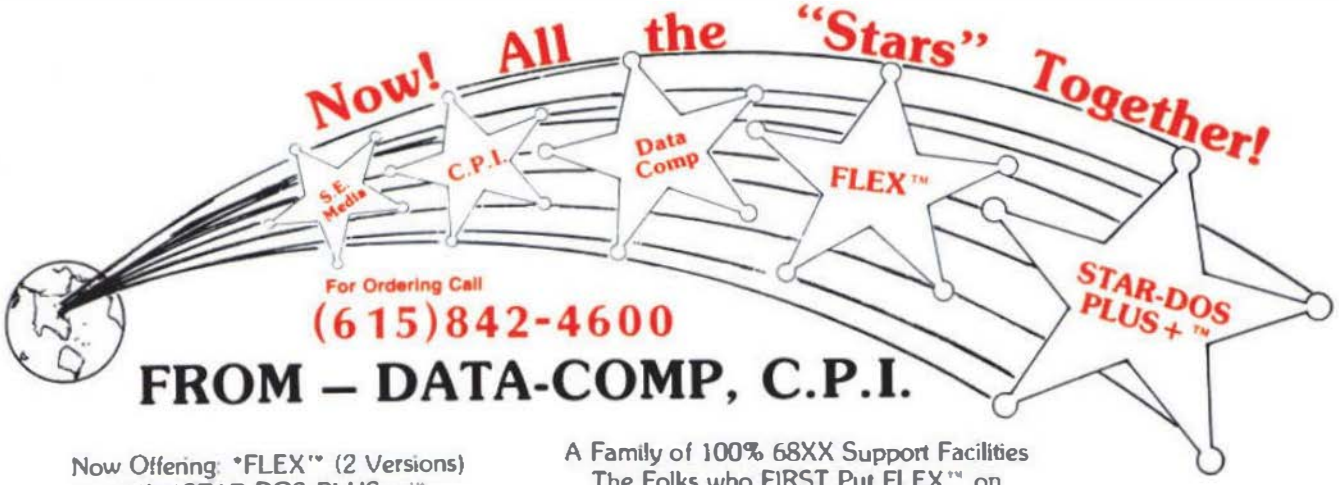
GMX products are not for consumer use. GMX expressly disclaims all warranties on any of its products which may be included in any product normally used for personal or family purposes.

Contact GMX by mail at 1337 West 37th Place, Chicago, IL 60609; or phone at (312) 927-5510; if your product is defective to arrange for its repair or replacement under this warranty.

Repair charges for GIMIX products after warranty period will be $35.00 per hour per board (minimum $35.00) plus parts. Customer pays freight charges both ways. If GIMIX determines that replacement is desirable instead, we will notify you. Charges for checking out complete system will be $500.00 plus parts, freight, and necessary board repairs.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

GIMIX inc. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055



FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY,CABINET,DISK DRIVE CABLE,J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS,RS-DOS OPERATING SYSTEMS.	$469.95
* Specify What CONTROLLER You Want J&M, or RADIO SHACK	
THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,P, AND COCO 11	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$325.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95
CABLES & OTHER INTERFACES CALL FOR PRICING	

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

MasterCard VISA

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

Introducing

S - 50 BUS / 68XX

Board and/or Computer
Terminals-CRTs-Printers
Disk Drives-etc.

REPAIRS

This

Not This

NOW AVAILABLE TO ALL S50/68XX USERS

The Data-Comp Division of CPI is proud to announce the availability of their service department facilities to **'ALL'** S50 Bus and 68XX users. Including all brands, SWTPC - GIMIX - SSB - HELIX and others, including the single board computers. *Please note that kit-built components are a special case, and will be handled on an individual basis, if accepted.

1. If you require service, the first thing you need to do is call the number below and describe your problem and *confirm a Data-Comp service & shipping number*! This is very important, Data-Comp will not accept or repair items not displaying this number! Also we cannot advise or help you troubleshoot on the telephone, we can give you a shipping number, but **NO** advice! Sorry!

2. All service shipments must include both a minimum $40.00 estimate/repair charge and pre-paid return shipping charges (should be same amount you pay to ship to Data-Comp).

3. If you desire a telephone estimate after your repair item is received, include an additional $5.00 to cover long distance charges. Otherwise an estimate will be mailed to you, if you requested an estimate. Estimates *must be requested*. Mailed estimates slow down the process considerably. However, if repairs are not desired, after the estimate is given, the $40.00 shall constitute the estimate charge, and the item(s) will be returned unrepaired providing sufficient return shipping charges were included with the item to be serviced. Please note that estimates are given in dollar amounts only.

4. Data-Comp service is the oldest and most experienced general S50/68XX service department in the world. We have over $100,000.00 in parts in stock. We have the most complete set of service documents for the various S50/68XX systems of anyone - **YET, WE DO NOT HAVE EVERYTHING!** But we sure have more than anyone else. We repair about 90% of all items we receive. Call for additional information or shipping instructions.

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615)842-4607
Telex 5106006630